

Guida per gli sviluppatori per la versione 2.x

# AWS SDK for Java 2.x



# AWS SDK for Java 2.x: Guida per gli sviluppatori per la versione 2.x

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

I marchi e l'immagine commerciale di Amazon non possono essere utilizzati in relazione a prodotti o servizi che non siano di Amazon, in una qualsiasi modalità che possa causare confusione tra i clienti o in una qualsiasi modalità che denigri o discrediti Amazon. Tutti gli altri marchi non di proprietà di Amazon sono di proprietà delle rispettive aziende, che possono o meno essere associate, collegate o sponsorizzate da Amazon.

---

# Table of Contents

Guida per gli sviluppatori - AWS SDK for Java 2.x .....	1
Inizia a usare l'SDK .....	1
Sviluppa applicazioni mobili .....	1
Manutenzione e supporto per le versioni principali dell'SDK .....	1
Risorse aggiuntive .....	2
Contribuisci all'SDK .....	2
Guida introduttiva .....	3
Fase 1: Configurazione per questo tutorial .....	3
Fase 2: Creare il progetto .....	3
Passaggio 3: scrivere il codice .....	8
Fase 4: Compilare ed eseguire l'applicazione .....	12
Riuscito .....	13
Rimozione .....	13
Passaggi successivi .....	13
Installazione .....	15
Panoramica della configurazione .....	15
Configura l'autenticazione .....	16
Configurazione per l'accesso Single Sign-On per l'SDK .....	16
Accedi utilizzando il AWS CLI .....	17
Installa Java e uno strumento di compilazione .....	18
Opzioni di autenticazione aggiuntive .....	18
Configurare un progetto Apache Maven .....	18
Prerequisiti .....	19
Creare un progetto Maven .....	19
Configurare il compilatore Java per Maven .....	20
Dichiarare l'SDK come dipendenza .....	21
Impostare le dipendenze per i moduli SDK .....	22
Creare il progetto .....	24
Configura un progetto Gradle .....	25
Configura un progetto GraalVM Native Image .....	31
Prerequisiti .....	31
Crea un progetto tramite modello tramite modello tramite CLI .....	32
Crea un'immagine nativa .....	32
Usa l'SDK .....	34

Collabora con i clienti del servizio .....	34
Crea un client di servizio .....	34
Configurazione client predefinita .....	35
Configurare i client di servizio .....	35
Effettua richieste .....	36
Gestisci le risposte .....	36
Chiudi il client del servizio .....	37
Gestisci le eccezioni .....	38
Usa i camerieri .....	39
Client HTTP .....	39
Tentativi .....	40
Timeout .....	40
Intercettori di esecuzione .....	41
Informazioni aggiuntive .....	41
Fornisci credenziali temporanee all'SDK .....	41
Configura l'accesso alle credenziali temporanee .....	42
Catena di fornitori di credenziali predefinita .....	43
Utilizza un fornitore o una catena di fornitori di credenziali specifico .....	45
Usa i profili .....	46
Carica credenziali temporanee da un processo esterno .....	49
Fornisci credenziali temporanee nel codice .....	52
Leggi le credenziali del ruolo IAM su Amazon EC2 .....	55
Usa Regioni AWS .....	57
configura esplicitamente un Regione AWS .....	57
Determina la regione dall'ambiente .....	58
Verifica la disponibilità del servizio .....	59
Scegli un endpoint specifico .....	60
Riduci i tempi di avvio dell'SDK per AWS Lambda .....	60
Utilizzate un client AWS HTTP basato su CRT .....	60
Rimuovi le dipendenze inutilizzate del client HTTP .....	61
Configura i client di servizio per abbreviare le ricerche .....	62
Inizializza il client SDK all'esterno del gestore di funzioni Lambda .....	63
Riduci al minimo l'iniezione di dipendenza .....	63
Usa un targeting Maven Archetype AWS Lambda .....	64
Lambda SnapStart .....	64
Modifiche alla versione 2.x che influiscono sul tempo di avvio .....	64



Risorse aggiuntive .....	64
Client HTTP .....	65
Client disponibili .....	65
Consigli per i clienti .....	66
Impostazioni predefinite intelligenti .....	69
Configurare il client HTTP basato su Apache .....	71
Configurare il client HTTP basato su URLConnection .....	76
Configurazione del client HTTP basato su Netty .....	82
Configurazione AWS dei client HTTP basati su CRT .....	88
Configurazione dei proxy HTTP .....	100
Gestione dell'eccezione .....	105
Perché le eccezioni non controllate? .....	105
AwsServiceException (e sottoclassi) .....	106
SdkClientException .....	107
Eccezioni e comportamenti relativi ai tentativi .....	107
Registrazione .....	107
File di configurazione Log4j 2 .....	108
Aggiungi dipendenza dalla registrazione .....	108
Errori e avvisi specifici dell'SDK .....	109
Registrazione riassuntiva di richieste/risposte .....	110
Registrazione SDK a livello di debug .....	111
Abilita la registrazione dei cavi .....	113
Imposta il TTL JVM per le ricerche dei nomi DNS .....	118
Come impostare il TTL JVM .....	119
Best practice .....	119
Riutilizza un client SDK .....	120
Chiudi i flussi di input .....	120
Ottimizza le configurazioni HTTP .....	120
Usa OpenSSL per Netty .....	121
Configura i timeout delle API .....	121
Usa le metriche .....	122
Risoluzione dei problemi .....	123
Ripristino della connessione .....	123
Timeout di connessione .....	123
Tempo di lettura scaduto .....	124
Timeout del pool di connessioni .....	125

Errori del percorso di classe .....	128
Errori di firma .....	129
Chiusura del pool di connessioni .....	131
Usa le funzionalità dell'SDK .....	133
Caratteristiche generali .....	133
Funzionalità specifiche del servizio .....	133
Lavora con risultati impaginati .....	133
Impaginazione sincrona .....	134
Impaginazione asincrona .....	137
Sondaggio sugli stati delle risorse .....	142
Prerequisiti .....	142
Usare i camerieri .....	143
Configura i camerieri .....	144
Esempi di codice .....	145
Usa la programmazione asincrona .....	145
Operazioni non in streaming .....	145
Operazioni di streaming .....	148
Operazioni avanzate .....	152
Lavora con HTTP/2 .....	153
Usa le metriche SDK .....	154
Prerequisiti .....	154
Come abilitare la raccolta delle metriche .....	155
Personalizza l'editore di metriche .....	156
Quando sono disponibili le metriche? .....	158
Quali informazioni vengono raccolte? .....	158
Come posso usare queste informazioni? .....	159
Metriche dei client di assistenza .....	159
Lavora con Servizi AWS .....	164
CloudWatch .....	164
Ottieni metriche da CloudWatch .....	165
Pubblica dati metrici personalizzati su CloudWatch .....	167
Lavora con gli CloudWatch allarmi .....	169
Usa Amazon CloudWatch Events .....	173
AWS servizi di database .....	177
Amazon DynamoDB .....	177
Amazon RDS .....	178

Amazon Redshift .....	178
Amazon Aurora senza server v1 .....	178
Amazon DocumentDB .....	179
DynamoDB .....	179
Lavora con le tabelle in DynamoDB .....	179
Lavora con gli elementi in DynamoDB .....	189
Mappa gli oggetti sugli elementi di DynamoDB .....	197
Amazon EC2 .....	316
Gestione delle Amazon EC2 istanze .....	316
Zone Regioni AWS di utilizzo e disponibilità .....	323
Collabora con i gruppi di sicurezza in Amazon EC2 .....	326
Utilizzare i metadati delle istanze Amazon EC2. ....	331
IAM .....	337
Gestione delle chiavi di IAM accesso .....	338
Gestisci IAM gli utenti .....	344
Crea politiche IAM .....	348
Lavora con IAM le politiche .....	356
Lavora con i certificati IAM del server .....	362
Kinesis .....	367
Sottoscrizione ad Amazon Kinesis Data Streams .....	367
Lambda .....	378
Invocazione di una funzione Lambda. ....	378
Elenca le funzioni Lambda .....	380
Eliminazione di una funzione Lambda .....	381
Amazon S3 .....	381
Utilizza punti di accesso o punti di accesso multiregionali .....	382
Operazioni relative ai bucket .....	383
Operazioni sugli oggetti .....	391
URL prefirmati .....	401
Accesso interregionale .....	409
Checksum .....	410
Usa un client S3 performante .....	412
Trasferisci file e directory .....	415
Amazon SNS .....	423
Creazione di un argomento .....	423
Elenca i tuoi Amazon SNS argomenti .....	424

Sottoscrizione di un endpoint a un argomento .....	425
Pubblicazione di un messaggio in un argomento .....	426
Annullamento della sottoscrizione di un endpoint da un argomento .....	427
Eliminazione di un argomento .....	428
Amazon SQS .....	429
Operazioni di coda .....	429
Operazioni relative ai messaggi .....	432
Amazon Transcribe .....	436
Configura il microfono .....	436
Crea un editore .....	437
Crea il client e avvia lo streaming .....	439
Ulteriori informazioni .....	435
Esempi di codice .....	442
Azioni e scenari .....	442
API Gateway .....	444
Application Auto Scaling .....	448
Application Recovery Controller .....	457
Aurora .....	460
Auto Scaling .....	495
Amazon Bedrock .....	557
Runtime di Amazon Bedrock .....	563
CloudFront .....	642
CloudWatch .....	662
CloudWatch Eventi .....	713
CloudWatch Registri .....	719
Amazon Cognito Identity .....	729
Provider di identità Amazon Cognito .....	737
Amazon Comprehend .....	764
DynamoDB .....	775
Amazon EC2 .....	853
Amazon ECS .....	925
Elastic Load Balancing - Versione 2 .....	939
MediaStore .....	984
OpenSearch Servizio .....	999
EventBridge .....	1007
Previsione .....	1039

---

AWS Glue .....	1052
HealthImaging .....	1076
IAM .....	1104
AWS IoT .....	1188
AWS IoT data .....	1215
Amazon Keyspaces .....	1217
Kinesis .....	1244
AWS KMS .....	1257
Lambda .....	1293
MediaConvert .....	1319
Migration Hub .....	1341
Amazon Personalize .....	1354
Eventi di Amazon Personalize .....	1384
Runtime di Amazon Personalize .....	1388
Amazon Pinpoint .....	1392
API SMS e Voce di Amazon Pinpoint .....	1437
Amazon Polly .....	1440
Amazon RDS .....	1446
Amazon Redshift .....	1488
Amazon Rekognition .....	1513
Registrazione del dominio Route 53 .....	1581
Amazon S3 .....	1604
S3 Glacier .....	1745
SageMaker .....	1761
Secrets Manager .....	1790
Amazon SES .....	1792
API Amazon SES v2 .....	1805
Amazon SNS .....	1823
Amazon SQS .....	1871
Step Functions .....	1892
AWS STS .....	1916
AWS Support .....	1919
Systems Manager .....	1942
Amazon Textract .....	1972
Amazon Transcribe .....	1983
Esempi di servizi incrociati .....	1999

Costruisci un'app per inviare dati a una tabella DynamoDB .....	2000
Creazione di un chatbot Amazon Lex .....	2000
Costruzione di un'applicazione Amazon SNS .....	2001
Crea un'applicazione di messaggistica .....	2001
Creazione di un'applicazione serverless per gestire foto .....	2002
Creazione di un'applicazione Web per tracciare i dati DynamoDB .....	2002
Creazione di un'applicazione Web per tracciare i dati Amazon Redshift .....	2003
Creazione di un tracciatore di elementi di lavoro di Aurora Serverless .....	2003
Crea un'applicazione per analizzare il feedback dei clienti .....	2004
Rilevamento dei DPI nelle immagini .....	2004
Rilevamento di oggetti nelle immagini .....	2005
Rilevamento di persone e oggetti in un video .....	2005
Monitora le prestazioni di DynamoDB .....	2006
Pubblicazione di messaggi nelle code .....	2006
Utilizzo di un'API Gateway per richiamare una funzione Lambda .....	2007
Utilizzo di Step Functions per richiamare le funzioni Lambda .....	2007
Utilizzo degli eventi pianificati per richiamare una funzione Lambda .....	2008
Sicurezza .....	2009
Protezione dei dati .....	2009
Transport Layer Security (TLS) .....	2010
Controlla le versioni TLS .....	2011
Applica le versioni TLS .....	2012
Migrazione a TLS 1.2 .....	2012
Identity and Access Management .....	2012
Destinatari .....	2013
Autenticazione con identità .....	2013
Gestione dell'accesso con policy .....	2017
Come Servizi AWS lavorare con IAM .....	2019
Risoluzione dei problemi di AWS identità e accesso .....	2020
Convalida della conformità .....	2022
Resilienza .....	2023
Sicurezza dell'infrastruttura .....	2024
Migrare alla versione 2 .....	2025
Novità della versione 2 .....	2025
tep-by-step Istruzioni S .....	2026
Panoramica delle fasi .....	2026

Esempio di migrazione .....	2027
Dal nome del pacchetto alle mappature ArtifactID .....	2038
Convenzione di denominazione .....	2039
Eccezioni .....	2039
Cosa c'è di diverso tra 1.x e 2.x .....	2044
Modifica del nome del pacchetto .....	2044
Aggiungere la versione 2.x al progetto .....	2045
POJO non modificabili .....	2046
Metodi Setter e Getter .....	2046
nomi delle classi del modello .....	2047
Biblioteche e servizi di pubblica utilità .....	2047
Modifiche al client .....	2049
Modifiche al provider delle credenziali .....	2094
Modifiche alla regione .....	2102
Modifiche alle operazioni, alle richieste e alle risposte .....	2104
Modifiche alle eccezioni .....	2105
Modifiche alla serializzazione .....	2106
Modifiche specifiche del servizio .....	2107
Modifiche al file di profilo .....	2113
Configurazione esterna .....	2114
Waiter .....	2117
S3 Transfer Manager .....	2121
Utilità per i metadati EC2 .....	2128
CloudFront preassegnazione .....	2135
Analisi degli URI S3 .....	2139
API IAM Policy Builder .....	2142
Uso SDK for Java 1.x side-by-side .....	2148
Chiave OpenPGP .....	2150
Chiave attuale .....	2150
Cronologia dei documenti .....	2152
.....	mmclx

# Guida per gli sviluppatori - AWS SDK for Java 2.x

L'AWS SDK for Java fornisce un'API Java per Servizi AWS. Utilizzando l'SDK, puoi creare applicazioni Java che funzionano con Amazon S3, Amazon EC2, Amazon DynamoDB, e altro ancora.

La AWS SDK for Java 2.x è una riscrittura importante del codice base della versione 1.x. È stata sviluppata su base Java 8+ e aggiunge diverse caratteristiche richieste frequentemente. Queste includono il supporto per l'I/O non bloccante e la possibilità di collegare un'implementazione HTTP diversa in fase di esecuzione.

Regolarmente, aggiungiamo ad AWS SDK for Java il supporto per nuovi servizi. Per un elenco delle modifiche e delle caratteristiche in una particolare versione, visualizza il [change log](#).

## Inizia a usare l'SDK

Se sei pronto a provare l'SDK, segui il tutorial. [Guida introduttiva](#)

Per configurare il tuo ambiente di sviluppo, consulta. [Installazione](#)

Se attualmente utilizzi la versione 1.x di SDK for Java, consulta [Migrare alla versione 2 per indicazioni](#) specifiche.

Per informazioni su come effettuare richieste a Amazon S3, Amazon EC2 e altro Amazon DynamoDB Servizi AWS, consulta [Use the SDK for Java](#) and [Work](#) with. Servizi AWS

## Sviluppa applicazioni mobili

Se sei uno sviluppatore di app per dispositivi mobili, Amazon Web Services fornisce il [AWS Amplify](#) framework.

## Manutenzione e supporto per le versioni principali dell'SDK

Per informazioni sulla manutenzione e il supporto per le versioni principali dell'SDK e le relative dipendenze sottostanti, consulta i seguenti argomenti nella Guida di riferimento agli [AWS SDK e agli strumenti](#):

- [AWS Politica di manutenzione degli SDK e degli strumenti](#)



- [AWSMatrice di supporto delle versioni degli SDK e degli strumenti](#)

## Risorse aggiuntive

Oltre a questa guida, di seguito sono elencate altre risorse online preziose per sviluppatori AWS SDK for Java.

- [AWS SDK for JavaRiferimento all'API 2.x](#)
- [Blog per gli sviluppatori Java](#)
- [Argomento sullo sviluppo di Java in AWS re:Post](#)
- [Sorgente SDK attiva](#) GitHub
- [AWSLibreria di esempi di codice SDK](#)
- [@awsforjava](#) (Twitter)

## Contribuisci all'SDK

Gli sviluppatori possono anche contribuire con commenti tramite i seguenti canali:

- Invia problemi su GitHub:
  - [Invia problemi relativi alla documentazione della Developer Guide](#)
  - [Inviando problemi SDK](#)
- [Partecipa a una chat informale sull'SDK sul canale gitter AWS SDK for Java 2.x](#)

# Inizia con la versione AWS SDK for Java 2.x

AWS SDK for Java 2.x Fornisce API Java per Amazon Web Services (AWS). Utilizzando l'SDK, è possibile creare applicazioni Java che funzionano con Amazon S3, Amazon EC2 DynamoDB, e altro ancora.

Questo tutorial mostra come utilizzare [Apache Maven](#) per definire le dipendenze per l'SDK for Java 2.x e quindi scrivere codice che si connette a per caricare un file. Amazon S3

Segui questi passaggi per completare questo tutorial:

- [Fase 1: Configurazione per questo tutorial](#)
- [Fase 2: Creare il progetto](#)
- [Fase 3: Scrivere il codice](#)
- [Fase 4: Compilare ed eseguire l'applicazione](#)

## Fase 1: Configurazione per questo tutorial

Prima di iniziare questo tutorial, è necessario quanto segue:

- Autorizzazione all'accesso Amazon S3
- Un ambiente di sviluppo Java configurato per accedere Servizi AWS tramite Single Sign-On a AWS IAM Identity Center

Usa le istruzioni riportate in questa guida [???](#) per prepararti a questo tutorial. Dopo aver [configurato l'ambiente di sviluppo con l'accesso Single Sign-On per Java SDK e aver avuto una sessione attiva sul portale di AWS accesso](#), continua con la Fase 2 di questo tutorial.

## Fase 2: Creare il progetto

Per creare il progetto per questo tutorial, esegui un comando Maven che richiede input su come configurare il progetto. Dopo aver inserito e confermato tutti gli input, Maven completa la creazione del progetto creando un `pom.xml` file Java stub.

1. Apri un terminale o una finestra del prompt dei comandi e accedi a una directory a tua scelta, ad esempio la tua cartella o. Desktop Home

2. Immettete il seguente comando nel terminale e Enter premete.

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

3. Immettete il valore elencato nella seconda colonna per ogni prompt.

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

4. Dopo aver inserito l'ultimo valore, Maven elenca le scelte effettuate. Conferma inserendo Yo reinserisci i valori inserendo. N

Maven crea la cartella di progetto denominata `getstarted` in base al `artifactId` valore inserito. All'interno della `getstarted` cartella, trova un `README.md` file che puoi rivedere, un `pom.xml` file e una `src` directory.

Maven crea il seguente albero di cartelle.

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

10 directories, 7 files

Quanto segue mostra il contenuto del file di progetto. `pom.xml`

## **pom.xml**

La `dependencyManagement` sezione contiene una dipendenza da AWS SDK for Java 2.x e la `dependencies` sezione ha una dipendenza per Amazon S3. Il progetto utilizza Java 1.8 a causa del 1.8 valore delle `maven.compiler.source` proprietà `and` `maven.compiler.target`

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

  <groupId>org.example</groupId>
  <artifactId>getstarted</artifactId>
```

```

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

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

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

  <dependency>

```

```

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

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

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

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

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

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

```

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

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

## Passaggio 3: scrivere il codice

Il codice seguente mostra la `App` classe creata da Maven. Il `main` metodo è il punto di ingresso nell'applicazione, che crea un'istanza della `Handler` classe e quindi ne `sendRequest` chiama il metodo.

### Classe **App**

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

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

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

        Handler handler = new Handler();
        handler.sendRequest();
    }
}
```

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

La `DependencyFactory` classe creata da Maven contiene il metodo `s3Client` factory che crea e restituisce un'istanza. [S3Client](#) L'`S3Client` istanza utilizza un'istanza del client HTTP basato su Apache. Questo perché hai specificato `apache-client` quando Maven ti ha chiesto quale client HTTP usare.

`DependencyFactory` È mostrato nel codice seguente.

## Classe `DependencyFactory`

```
package org.example;

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

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

    private DependencyFactory() {}

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

La `Handler` classe contiene la logica principale del programma. Quando `Handler` viene creata un'istanza di nella `App` classe, `DependencyFactory` fornisce il `S3Client` servizio client. Il codice utilizza l'`S3Client` istanza per chiamare il servizio Amazon S3.

Maven genera la seguente `Handler` classe con un commento. *TODO* Il passaggio successivo del tutorial sostituisce il *TODO* codice with.



## Handler classe, generata da Maven

```
package org.example;

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

public class Handler {
    private final S3Client s3Client;

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

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

Per compilare la logica, sostituisci l'intero contenuto della `Handler` classe con il codice seguente. Il `sendRequest` metodo viene compilato e vengono aggiunte le importazioni necessarie.

## Handler classe, implementata

Il codice crea innanzitutto un nuovo bucket S3 con l'ultima parte del nome generata utilizzando `System.currentTimeMillis()` per rendere unico il nome del bucket.

Dopo aver creato il bucket nel `createBucket()` metodo, il programma carica un oggetto utilizzando il metodo di [putObject](#) `S3Client`. Il contenuto dell'oggetto è una semplice stringa creata con il `RequestBody.fromString` metodo.

Infine, il programma elimina l'oggetto seguito dal bucket nel `cleanup` metodo.

```
package org.example;

import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.CreateBucketRequest;
import software.amazon.awssdk.services.s3.model.DeleteBucketRequest;
import software.amazon.awssdk.services.s3.model.DeleteObjectRequest;
import software.amazon.awssdk.services.s3.model.HeadBucketRequest;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
```

```
import software.amazon.awssdk.services.s3.model.S3Exception;

public class Handler {
    private final S3Client s3Client;

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

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

        createBucket(s3Client, bucket);

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

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

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

        cleanUp(s3Client, bucket, key);

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

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

```
        System.out.printf("%n");
    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

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

## Fase 4: Compilare ed eseguire l'applicazione

Dopo che il progetto è stato creato e contiene la Handler classe completa, create ed eseguite l'applicazione.

1. Assicurati di avere una sessione IAM Identity Center attiva. Per farlo, esegui il AWS Command Line Interface comando `aws sts get-caller-identity` e controlla la risposta. Se non hai una sessione attiva, consulta [questa sezione](#) per le istruzioni.
2. Apri un terminale o una finestra del prompt dei comandi e accedi alla directory `getstarted` del progetto.
3. Usa il seguente comando per creare il tuo progetto:

```
mvn clean package
```

4. Utilizzate il seguente comando per eseguire l'applicazione.

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

Per visualizzare il nuovo bucket e l'oggetto creati dal programma, effettuate le seguenti operazioni.

1. In `InHandler.java`, commenta la riga `cleanup(s3Client, bucket, key)` del `sendRequest` metodo e salva il file.
2. Ricostruisci il progetto `mvn clean package` eseguendo.
3. Esegui nuovamente `mvn exec:java -Dexec.mainClass="org.example.App"` per caricare l'oggetto di testo ancora una volta.
4. Accedi [alla console S3](#) per visualizzare il nuovo oggetto nel bucket appena creato.

Dopo aver visualizzato il file, elimina l'oggetto e quindi elimina il bucket.

## Riuscito

Se il tuo progetto Maven è stato creato ed eseguito senza errori, allora congratulazioni! Hai creato con successo la tua prima applicazione Java utilizzando l'SDK for Java 2.x.

## Rimozione

Per ripulire le risorse create durante questo tutorial, procedi come segue:

- Se non l'hai già fatto, [nella console S3](#), elimina tutti gli oggetti e i bucket creati durante l'esecuzione dell'applicazione.
- Elimina la cartella del progetto `()getstarted`.

## Passaggi successivi

Ora che hai le nozioni di base, puoi scoprire quanto segue:

- [Lavorare con Amazon S3](#)

- [Lavorare con altri Amazon Web Services](#), ad esempio [DynamoDB](#) [Amazon EC2](#), e [vari servizi di database](#)
- [Usa l'SDK](#)
- [Sicurezza per AWS SDK for Java](#)

# Configura il AWS SDK for Java 2.x

Questa sezione fornisce informazioni su come configurare l'ambiente di sviluppo e i progetti per l'utilizzo di AWS SDK for Java 2.x.

## Panoramica della configurazione

Per sviluppare correttamente applicazioni che accedono Servizi AWS tramite AWS SDK for Java, sono necessarie le seguenti condizioni:

- L'SDK Java deve avere accesso alle credenziali per [autenticare le richieste](#) per conto dell'utente.
- Le [autorizzazioni del ruolo IAM](#) configurato per l'SDK devono consentire l'accesso a ciò che l'Servizio AWS applicazione richiede. Le autorizzazioni associate alla policy PowerUserAccess AWS gestita sono sufficienti per la maggior parte delle esigenze di sviluppo.
- Un ambiente di sviluppo con i seguenti elementi:
  - [File di configurazione condivisi](#) configurati in almeno uno dei seguenti modi:
    - Il config file contiene le [impostazioni Single Sign-On di IAM Identity Center in](#) modo che l'SDK possa ottenere le credenziali. AWS
    - Il `credentials` file contiene credenziali temporanee.
  - Un'[installazione di Java 8](#) o versione successiva.
  - [Uno strumento di automazione delle build come Maven o Gradle.](#)
  - Un editor di testo per lavorare con il codice.
  - (Facoltativo, ma consigliato) Un IDE (ambiente di sviluppo integrato) come [IntelliJ IDEA](#), [Eclipse](#) o [NetBeans](#)

Quando usi un IDE, puoi anche integrarlo con Kit di strumenti AWS s per lavorare più facilmente. Servizi AWS I [Kit di strumenti AWS per IntelliJ](#) e [AWS Toolkit for Eclipse](#) sono due toolkit che è possibile utilizzare per lo sviluppo in Java.

- Una sessione attiva del portale di AWS accesso quando si è pronti per eseguire l'applicazione. La usi AWS Command Line Interface per [avviare la procedura di accesso al portale di accesso](#) di IAM Identity Center. AWS

### ⚠ Important

Le istruzioni in questa sezione di configurazione presuppongono che tu o l'organizzazione utilizzi IAM Identity Center. Se la tua organizzazione utilizza un provider di identità esterno che funziona indipendentemente da IAM Identity Center, scopri come ottenere credenziali temporanee da utilizzare con l'SDK for Java. Segui [queste istruzioni](#) per aggiungere credenziali temporanee al file. `~/.aws/credentials`

Se il tuo provider di identità aggiunge automaticamente credenziali temporanee al `~/.aws/credentials` file, assicurati che il nome del profilo sia `[default]` tale da non dover fornire un nome di profilo all'SDK o. AWS CLI

## Configura l'autenticazione

L'argomento [Autenticazione e accesso](#) nella Guida di riferimento agli AWS SDK e agli strumenti descrive le diverse opzioni di autenticazione. Ti consigliamo di seguire le istruzioni per [configurare l'accesso a IAM Identity Center in modo che l'SDK possa acquisire le credenziali](#). Dopo aver seguito le istruzioni, [il sistema è configurato](#) per consentire all'SDK di autenticare le richieste.

## Configurazione per l'accesso Single Sign-On per l'SDK

Dopo aver completato il passaggio 2 nella [sezione sull'accesso programmatico](#) in modo che l'SDK possa utilizzare l'autenticazione IAM Identity Center, il sistema dovrebbe contenere i seguenti elementi.

- Il AWS CLI, che viene utilizzato per avviare una [sessione del portale di AWS accesso](#) prima di eseguire l'applicazione.
- Un `~/.aws/config` file che contiene un [profilo predefinito](#). L'SDK for Java utilizza la configurazione del provider di token SSO del profilo per acquisire le credenziali prima di inviare richieste a. AWS Il `sso_role_name` valore, che è un ruolo IAM connesso a un set di autorizzazioni IAM Identity Center, dovrebbe consentire l'accesso ai dati Servizi AWS utilizzati nell'applicazione.

Il seguente `config` file di esempio mostra un profilo predefinito impostato con la configurazione del provider di token SSO. L'impostazione `sso_session` del profilo si riferisce alla sezione `sso-session` denominata. La `sso-session` sezione contiene le impostazioni per avviare una sessione del portale di AWS accesso.

```
[default]
sso_session = my-sso
sso_account_id = 111122223333
sso_role_name = SampleRole
region = us-east-1
output = json

[sso-session my-sso]
sso_region = us-east-1
sso_start_url = https://provided-domain.awsapps.com/start
sso_registration_scopes = sso:account:access
```

Per maggiori dettagli sulle impostazioni utilizzate nella configurazione del provider di token SSO, consulta Configurazione del provider di [token SSO](#) nella Guida di riferimento agli AWS SDK e agli strumenti.

Se il tuo ambiente di sviluppo non è configurato per l'accesso programmatico come mostrato in precedenza, segui il [passaggio 2 della Guida di riferimento agli SDK](#).

## Accedi utilizzando il AWS CLI

Prima di eseguire un'applicazione che consente l'accesso Servizi AWS, è necessaria una sessione attiva del portale di AWS accesso affinché l'SDK utilizzi l'autenticazione IAM Identity Center per risolvere le credenziali. Esegui il seguente comando in AWS CLI per accedere al portale di accesso.

```
aws sso login
```

Poiché disponi di una configurazione predefinita del profilo, non devi chiamare il comando con un'opzione `--profile`. Se la configurazione del provider di token SSO utilizza un profilo denominato, il comando è `aws sso login --profile named-profile`.

Per verificare se hai già una sessione attiva, esegui il AWS CLI comando seguente.

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

La risposta a questo comando dovrebbe restituire l'account IAM Identity Center e il set di autorizzazioni configurati nel file `config` condiviso.



### Note

Se hai già una sessione attiva del portale di AWS accesso ed esegui `aws sso login`, non ti verrà richiesto di fornire credenziali.

Tuttavia, verrà visualizzata una finestra di dialogo che richiede l'autorizzazione per accedere botocore alle informazioni. botocore è la base per AWS CLI .

Seleziona Consenti per autorizzare l'accesso alle tue informazioni per AWS CLI e SDK for Java.

## Installa Java e uno strumento di compilazione

Il tuo ambiente di sviluppo richiede quanto segue:

- Java 8 o versione successiva. AWS SDK for Java [Funziona con l'Oracle Java SE Development Kit e con le distribuzioni di Open Java Development Kit \(OpenJDK\) come Red Hat OpenJDK e Amazon CorrettoAdoptium.](#)
- Uno strumento di compilazione o IDE che supporta Maven Central come Apache Maven, Gradle o IntelliJ.
  - [Per informazioni su come installare e utilizzare Maven, consulta https://maven.apache.org/.](https://maven.apache.org/)
  - [Per informazioni su come installare e usare Gradle, vedi https://gradle.org/.](https://gradle.org/)
  - [Per informazioni su come installare e utilizzare IntelliJ IDEA, vedere https://www.jetbrains.com/idea/.](https://www.jetbrains.com/idea/)

## Opzioni di autenticazione aggiuntive

Per ulteriori opzioni sull'autenticazione per l'SDK, come l'uso di profili e variabili di ambiente, consulta il capitolo sulla [configurazione](#) nella Guida di riferimento agli AWS SDK e agli strumenti.

## Configurare un progetto Apache Maven

[Puoi usare Apache Maven per configurare e creare AWS SDK for Java progetti o per creare l'SDK stesso.](#)

## Prerequisiti

Per utilizzare AWS SDK for Java con Maven, dovrai disporre di quanto segue:

- Java 8.0 o versione successiva. Puoi scaricare la versione più recente del software Java SE Development Kit da <http://www.oracle.com/technetwork/java/javase/downloads/>. AWS SDK for Java funziona anche con [OpenJDK](#) e Amazon Corretto una distribuzione di Open Java Development Kit (OpenJDK). Scarica la versione più recente di OpenJDK da <https://openjdk.java.net/install/index.html>. [Scarica l'ultima versione Amazon Corretto 8 o Amazon Corretto 11 dalla pagina. Corretto](#)
- Apache Maven. Se hai bisogno di installare Maven, vai su <http://maven.apache.org/> per scaricarlo e installarlo.

## Creare un progetto Maven

Per creare un progetto Maven dalla riga di comando, esegui il seguente comando da un terminale o da una finestra del prompt dei comandi.

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

### Note

Sostituisci `com.example.myapp` con lo spazio dei nomi del pacchetto completo dell'applicazione. Sostituisci anche `myapp` con il nome del tuo progetto. Questo sarà il nome della directory per il progetto.

[Per utilizzare la versione più recente dell'archetipo, sostituisci `2.X.X` con la versione più recente di Maven central.](#)

Questo comando crea un progetto Maven utilizzando l'archetype templating toolkit. L'archetipo genera lo scaffolding per un progetto di gestione delle funzioni. AWS Lambda Questo archetipo di progetto è preconfigurato per la compilazione con Java SE 8 e include una dipendenza dalla versione dell'SDK for Java 2.x specificata con. `-DarchetypeVersion`

Per ulteriori informazioni sulla creazione e la configurazione di progetti Maven, consulta la [Guida alle operazioni di base Maven](#).

## Configurare il compilatore Java per Maven

Se avete creato il progetto utilizzando l'archetipo del AWS Lambda progetto come descritto in precedenza, la configurazione del compilatore Java è già stata completata automaticamente.

Per verificare che questa configurazione sia presente, inizia aprendo il file `pom.xml` dalla cartella del progetto creata (ad esempio `myapp`) quando hai eseguito il comando precedente. Guarda le righe 11 e 12 per vedere l'impostazione della versione del compilatore Java per questo progetto Maven e l'inclusione richiesta del plugin del compilatore Maven sulle righe 71-75.

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

Se crei il tuo progetto con un archetipo diverso o utilizzando un altro metodo, devi assicurarti che il plugin del compilatore Maven faccia parte della build e che le sue proprietà di origine e destinazione siano entrambe impostate su 1.8 nel file `pom.xml`.

Vedi lo snippet precedente per una modalità di configurazione di tali impostazioni richieste.

In alternativa, puoi impostare la configurazione del compilatore in linea con la dichiarazione del plugin, come segue.

```
<project>
  <build>
```

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

## Dichiarare l'SDK come dipendenza

Per utilizzare AWS SDK for Java nel progetto, occorre dichiararla come una dipendenza nel file `pom.xml` del progetto.

Se avete creato il progetto utilizzando l'archetipo del progetto come descritto in precedenza, l'ultima versione dell'SDK è già configurata come dipendenza nel progetto.

L'archetipo genera una dipendenza dall'artefatto BOM (distinta dei materiali) per l'id del gruppo `software.amazon.awssdk`. Con un BOM, non è necessario specificare la versione Maven per le dipendenze dei singoli artefatti che condividono lo stesso ID di gruppo.

Se hai creato il tuo progetto Maven in un modo diverso, configura l'ultima versione dell'SDK per il tuo progetto assicurandoti che il file `pom.xml` contenga quanto segue.

```
<project>
  <properties>
    <aws.java.sdk.version>2.X.X</aws.java.sdk.version>
  </properties>
  <dependencyManagement>
    <dependencies>
      <dependency>
        <groupId>software.amazon.awssdk</groupId>
        <artifactId>bom</artifactId>
        <version>${aws.java.sdk.version}</version>
        <type>pom</type>
        <scope>import</scope>
      </dependency>
    </dependencies>
  </dependencyManagement>
</project>
```

```
</dependencyManagement>
</project>
```

### Note

Sostituisci **2.X.X** nel file con l'ultima versione di `dipom.xml`. [AWS SDK for Java 2.x](#)

## Impostare le dipendenze per i moduli SDK

Dopo aver configurato l'SDK, puoi aggiungere dipendenze per uno o più moduli AWS SDK for Java da utilizzare nel progetto.

Sebbene sia possibile specificare il numero di versione per ogni componente, non è necessario perché la versione dell'SDK è già stata dichiarata nella `dependencyManagement` sezione utilizzando l'elemento della distinta base. Per caricare una versione diversa di un determinato modulo, specificate un numero di versione per la sua dipendenza.

Se avete creato il progetto utilizzando l'archetipo del progetto come descritto in precedenza, il progetto è già configurato con dipendenze multiple. Queste includono le dipendenze per i gestori di AWS Lambda funzioni e Amazon S3, come segue.

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

    <dependency>
      <groupId>software.amazon.awssdk</groupId>
```

```
    <artifactId>url-connection-client</artifactId>
  </dependency>

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

### Note

Nell'`pom.xml` esempio precedente, le dipendenze provengono da diverse `s. groupId`. La `s3` dipendenza proviene da `software.amazon.awssdk`, mentre la `aws-lambda-java-core` dipendenza proviene da `com.amazonaws`. La configurazione della gestione delle dipendenze BOM influisce sugli artefatti `software.amazon.awssdk`, quindi è necessaria una versione per l'artefatto `aws-lambda-java-core`.

Per lo sviluppo di gestori di funzioni Lambda utilizzando l'SDK for Java `aws-lambda-java-core 2.x`, è la dipendenza corretta. Tuttavia, se l'applicazione deve gestire le risorse Lambda, l'utilizzo di operazioni come `listFunctions`, `deleteFunction`, `invokeFunction`, `createFunction`, e richiede la seguente dipendenza.

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

### Note

La `s3` dipendenza esclude le dipendenze e le dipendenze transitive. `netty-nio-client` `apache-client` [Al posto di uno di questi client HTTP, l'archetipo include la `url-connection-client` dipendenza, che aiuta a ridurre la latenza di avvio delle funzioni. AWS Lambda](#)

Aggiungi i moduli al tuo progetto per le Servizio AWS funzionalità necessarie al tuo progetto. I moduli (dipendenze) gestiti dalla AWS SDK for Java BOM sono elencati nell'archivio centrale di [Maven](#).

### Note

Puoi esaminare il file `pom.xml` da un esempio di codice per determinare di quali dipendenze hai bisogno per il tuo progetto. [Ad esempio, se sei interessato alle dipendenze per il servizio DynamoDB, guarda questo esempio dal AWS Code Examples Repository su GitHub \(Cerca il `pom.xml` file in `/javav2/example\_code/dynamodb`.\)](#)

## Creare l'intero SDK nel progetto

Per ottimizzare l'applicazione, ti consigliamo di inserire solo i componenti necessari anziché tutto l'SDK. Tuttavia, per costruire l'intero AWS SDK for Java nel tuo progetto, dichiaralo nel tuo file `pom.xml`, come segue.

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

## Creare il progetto

Dopo aver configurato il file `pom.xml`, puoi usare Maven per creare il tuo progetto.

Per creare il tuo progetto Maven dalla riga di comando, apri un terminale o una finestra del prompt dei comandi, vai alla directory del progetto (ad esempio `myapp`), inserisci o incolla il seguente comando, quindi premi Enter o Invio.

```
mvn package
```

Viene così creato un singolo file `.jar` (JAR) nella directory `target` (ad esempio, `myapp/target`). Questo file JAR contiene tutti i moduli SDK specificati come dipendenze nel file `pom.xml`.

# Configura un progetto Gradle

Puoi usare [Gradle](#) per configurare e creare progetti. AWS SDK for Java

I passaggi iniziali nell'esempio seguente provengono dalla [guida introduttiva di Gradle](#) per la versione 8.4. Se si utilizza una versione diversa, i risultati potrebbero differire leggermente.

Per creare un'applicazione Java con Gradle (riga di comando)

1. Crea una cartella per contenere il tuo progetto. In questo esempio, demo è il nome della directory.
2. All'interno della demo directory, esegui il `gradle init` comando e fornisci i valori evidenziati in rosso, come mostrato nel seguente output della riga di comando. Per la procedura dettagliata, scegliamo Kotlin come linguaggio DSL per lo script di compilazione, ma alla fine di questo argomento viene mostrato anche un esempio completo di Groovy.

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

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

Select implementation language:
1: C++
2: Groovy
3: Java
4: Kotlin
5: Scala
6: Swift
Enter selection (default: Java) [1..6] 3

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

Select test framework:
```



```

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

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

> Task :init
To learn more about Gradle by exploring our Samples at https://docs.gradle.org/8.4/
samples/sample_building_java_applications.html

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

```

3. Al termine dell'iniziativa, la demo directory contiene la seguente struttura ad albero. Diamo un'occhiata più da vicino al file di build principale, `build.gradle.kts` (evidenziato in rosso), nella prossima sezione.

```

### app
#   ### build.gradle.kts
#   ### src
#     ### main
#     #   ### java
#     #   #   ### demo
#     #   #       ### App.java
#     #   ### resources
#     ### test
#     ### java
#     #   ### demo
#     #       ### AppTest.java
#     ### resources
### gradle
#   ### wrapper
#     ### gradle-wrapper.jar
#     ### gradle-wrapper.properties
### gradlew
### gradlew.bat
### settings.gradle.kts

```

Il `build.gradle.kts` file contiene il seguente contenuto impalcaturato.

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

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

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

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

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

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

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

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

```
}

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

4. Usa il file di build Gradle impalcaturato come base per il tuo progetto. AWS
  - a. Per gestire le dipendenze SDK per il tuo progetto Gradle, aggiungi la distinta base (BOM) di Maven alla sezione del file. AWS SDK for Java 2.x dependencies build.gradle.kts

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

#### Note

In questo file di build di esempio, sostituisci 2.21.1 con l'ultima versione di SDK for Java 2.x. [Trova l'ultima versione disponibile nel repository centrale di Maven.](#)

- b. Specificate i moduli SDK necessari alla vostra applicazione nella sezione. dependencies. Ad esempio, quanto segue aggiunge una dipendenza da Amazon Simple Storage Service.

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

Gradle risolve automaticamente la versione corretta delle dipendenze dichiarate utilizzando le informazioni della BOM.

Gli esempi seguenti mostrano i file di build Gradle completi nelle DSL Kotlin e Groovy. Il file di build contiene dipendenze per Amazon S3, autenticazione, registrazione e test. La versione di origine e di destinazione di Java è la versione 11.

### Kotlin DSL (build.gradle.kts)

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

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

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

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

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

```
    }  
}  
  
application {  
    // Define the main class for the application.  
    mainClass.set("demo.App")  
}  
  
tasks.named<Test>("test") {  
    // Use JUnit Platform for unit tests.  
    useJUnitPlatform()  
}
```

## Groovy DSL (build.gradle)

```
/*  
 * This file was generated by the Gradle 'init' task.  
 *  
 * This generated file contains a sample Java application project to get you  
 * started.  
 * For more details on building Java & JVM projects, please refer to https://  
docs.gradle.org/8.4/userguide/building\_java\_projects.html in the Gradle  
 * documentation.  
 */  
  
plugins {  
    // Apply the application plugin to add support for building a CLI application in  
    Java.  
    id 'application'  
}  
  
repositories {  
    // Use Maven Central for resolving dependencies.  
    mavenCentral()  
}  
  
dependencies {  
    implementation platform('software.amazon.awssdk:bom:2.21.1')  
    implementation 'software.amazon.awssdk:s3'  
    implementation 'software.amazon.awssdk:sso'  
    implementation 'software.amazon.awssdk:ssoidc'  
    implementation platform('org.apache.logging.log4j:log4j-bom:2.20.0')  
    implementation 'org.apache.logging.log4j:log4j-slf4j2-impl'}
```

```
implementation 'org.apache.logging.log4j:log4j-1.2-api'  
testImplementation platform('org.junit:junit-bom:5.10.0')  
testImplementation 'org.junit.jupiter:junit-jupiter'  
}  
  
// Apply a specific Java toolchain to ease working on different environments.  
java {  
    toolchain {  
        languageVersion = JavaLanguageVersion.of(11)  
    }  
}  
  
application {  
    // Define the main class for the application.  
    mainClass = 'demo_groovy.App'  
}  
  
tasks.named('test') {  
    // Use JUnit Platform for unit tests.  
    useJUnitPlatform()  
}
```

Per i passaggi successivi, consulta la guida introduttiva sul sito Web di Gradle per istruzioni su come [creare ed eseguire un'applicazione Gradle](#).

## Configurare un progetto GraalVM Native Image per AWS SDK for Java

Con le versioni 2.16.1 e successive, AWS SDK for Java fornisce out-of-the-box supporto per le applicazioni GraalVM Native Image. Usa l'archetipo `archetype-app-quickstart` Maven per configurare un progetto con supporto nativo integrato per le immagini.

### Prerequisiti

- Completa i passaggi in [Configurazione dell'AWS SDK for Java 2.x](#).
- Installa l'[immagine nativa di GraalVM](#).

## Crea un progetto tramite modello tramite modello tramite CLI

Per creare un progetto Maven con supporto nativo integrato per le immagini, in un terminale o in una finestra del prompt dei comandi, usa il seguente comando.

### Note

Sostituisci `com.example.mynativeimageapp` con il namespace completo del pacchetto della tua applicazione. Inoltre, `mynativeimageapp` sostituisilo con il nome del progetto. Questo sarà il nome della directory per il progetto.

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

Questo comando crea un progetto Maven configurato con dipendenze per il `software.amazon.awssdk` per `Amazon S3`, e il `clientApacheHttpClient` HTTP. Include anche una dipendenza per il [plug-in GraalVM Native Image Maven](#), in modo da poter creare immagini native usando Maven.

Per includere le dipendenze per un altro `Amazon Web Services`, imposta il valore del `-Dservice` parametro sull'ID dell'articolo di quel servizio. Esempi includono `dynamodb`, `comprehend` e `pinpoint`. Per un elenco completo degli ID degli artefatti, consulta l'elenco delle dipendenze gestite per [software.amazon.awssdk su Maven Central](#).

Per utilizzare un client HTTP asincrono, impostate il `-DhttpClient` parametro su `sunetty-nio-client`. Per `URLConnectionHttpClient` utilizzarlo invece come client HTTP sincrono `apache-client`, imposta il `-DhttpClient` parametro su `url-connection-client`.

## Crea un'immagine nativa

Dopo aver creato il progetto, esegui il seguente comando dalla directory del progetto, ad esempio `mynativeimageapp`:

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

Questo crea un'applicazione di immagine nativa nellatarget directory, ad esempiotarget/  
mynativeimageapp.



# Usa il AWS SDK for Java 2.x

Dopo aver completato i passaggi descritti in [Configurazione dell'SDK](#), sei pronto per effettuare richieste a AWS servizi come Amazon S3, DynamoDB, IAM, Amazon EC2 e molti altri.

## Collabora con i clienti del servizio

### Crea un client di servizio

Per effettuare una richiesta a un Servizio AWS, devi prima creare un'istanza di un client di servizio per quel servizio utilizzando il metodo static factory, `builder()`. Il `builder()` metodo restituisce un `builder` oggetto che consente di personalizzare il client del servizio. I metodi setter fluent restituiscono l'oggetto `builder`. In questo modo puoi concatenare chiamate al metodo per comodità e per avere un codice più leggibile. Dopo aver configurato le proprietà desiderate, chiamate il `build()` metodo per creare il client.

Ad esempio, il seguente frammento di codice crea un'istanza di un `Ec2Client` oggetto come client di servizio per Amazon EC2.

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

#### Note

I client del servizio nell'SDK sono thread-safe. Per prestazioni ottimali, ti consigliamo di trattarli come oggetti di lunga durata. Ogni client dispone di una propria risorsa di pooling delle connessioni che viene rilasciata durante il processo di garbage collection del client. Un oggetto client di servizio è immutabile, quindi è necessario creare un nuovo client per ogni servizio a cui si effettuano richieste o se si desidera utilizzare una configurazione diversa per effettuare richieste allo stesso servizio.

Non è necessario specificare `Region` nel service client builder per tutti i AWS servizi; tuttavia, è consigliabile impostare la regione per le chiamate API effettuate nelle applicazioni. Per ulteriori informazioni, consulta la sezione [Selezione della AWS regione](#).

## Configurazione client predefinita

I generatori client dispongono di un altro metodo factory denominato `create()`. Questo metodo consente di creare un client del servizio con la configurazione predefinita. Utilizza la catena di provider predefinita per caricare le credenziali e il Regione AWS. Se non è possibile determinare le credenziali o la regione dall'ambiente in cui è in esecuzione l'applicazione, la chiamata a `create` ha esito negativo. Vedi [Utilizzo delle credenziali](#) e [Selezione della regione](#) per ulteriori informazioni su come l'SDK determina le credenziali e la regione da utilizzare.

Ad esempio, il seguente frammento di codice crea un'istanza di un `DynamoDbClient` oggetto come client di servizio per Amazon DynamoDB:

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

## Configurare i client di servizio

Per personalizzare la configurazione di un client di servizio, utilizzate i setter del metodo `builder()` factory. Per comodità e per creare codice più leggibile, concatenate i metodi per impostare più opzioni di configurazione.

L'esempio seguente mostra un `S3Client` file configurato con diverse impostazioni personalizzate.

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

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

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

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

## Effettua richieste

Utilizza il client di servizio per effettuare richieste al corrispondente Servizio AWS.

Ad esempio, questo frammento di codice mostra come creare un `RunInstancesRequest` oggetto per creare una nuova istanza Amazon EC2:

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

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

Anziché creare una richiesta e passarla all'istanza, l'SDK fornisce dei builder che puoi utilizzare per creare una richiesta. Con un builder puoi utilizzare un'espressione lambda Java per creare la richiesta «in linea».

L'esempio seguente riscrive l'esempio precedente utilizzando la versione del `runInstances` [metodo che utilizza un](#) builder per creare la richiesta.

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

## Gestisci le risposte

L'SDK restituisce un oggetto di risposta per la maggior parte delle operazioni di servizio. Il codice può elaborare le informazioni nell'oggetto di risposta in base alle vostre esigenze.

Ad esempio, il seguente frammento di codice stampa l'ID di prima istanza restituito con l'[RunInstancesResponse](#) oggetto dalla richiesta precedente.

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

Tuttavia, non tutte le operazioni restituiscono un oggetto di risposta con dati specifici del servizio. In queste situazioni, è possibile interrogare lo stato della risposta HTTP per sapere se l'operazione ha avuto esito positivo.

Ad esempio, il codice nello snippet seguente verifica la risposta HTTP per verificare se il [DeleteContactList](#) funzionamento di Amazon Simple Email Service ha avuto successo.

```
SesV2Client sesv2Client = SesV2Client.create();

DeleteContactListRequest request = DeleteContactListRequest.builder()
    .contactListName("ExampleContactListName")
    .build();

DeleteContactListResponse response = sesv2Client.deleteContactList(request);
if (response.sdkHttpResponse().isSuccessful()) {
    System.out.println("Contact list deleted successfully");
} else {
    System.out.println("Failed to delete contact list. Status code: " +
        response.sdkHttpResponse().statusCode());
}
```

## Chiudi il client del servizio

Come procedura ottimale, è consigliabile utilizzare un client di servizio per più chiamate al servizio API durante la vita di un'applicazione. Tuttavia, se hai bisogno di un client di servizio per un uso singolo o non hai più bisogno del client di servizio, chiudilo.

Richiamate il `close()` metodo quando il client di servizio non è più necessario per liberare risorse.

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

Se hai bisogno di un client di servizio da utilizzare una sola volta, puoi creare un'istanza del client di servizio come risorsa in un'istruzione `-with-resources`. I client di servizio implementano l'[Autoclosable](#) interfaccia, quindi il JDK chiama automaticamente il metodo alla fine dell'istruzione `close()`.

L'esempio seguente mostra come utilizzare un client di servizio per una chiamata una tantum. La chiamata `StsClient` che chiama AWS Security Token Service viene chiusa dopo la restituzione dell'ID dell'account.

```
import software.amazon.awssdk.services.sts.StsClient;

String getAccountID() {
    try (StsClient stsClient = StsClient.create()) {
        return stsClient.getCallerIdentity().account();
    }
}
```

## Gestisci le eccezioni

L'SDK utilizza eccezioni di runtime (o non controllate), che forniscono un controllo preciso sulla gestione degli errori e garantiscono che la gestione delle eccezioni sia scalabile in base all'applicazione.

Una [SdkServiceException](#) una delle sue sottoclassi è la forma di eccezione più comune generata dall'SDK. Queste eccezioni rappresentano le risposte del servizio. AWS Puoi anche gestire un [SdkClientException](#) problema sul lato client (ad esempio, nell'ambiente di sviluppo o applicativo), come un errore di connessione di rete.

Questo frammento di codice mostra un modo per gestire le eccezioni di servizio quando si carica un file su Amazon S3 Il codice di esempio rileva le eccezioni del client e del server, registra i dettagli ed esiste l'applicazione.

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

try {

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

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

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

Per ulteriori informazioni, vedere [Gestione delle eccezioni](#).

## Usa i camerieri

L'elaborazione di alcune richieste richiede tempo, ad esempio la creazione di una nuova tabella in DynamoDB o la creazione di un nuovo Amazon S3 bucket. Per assicurarti che la risorsa sia pronta prima che il codice continui a funzionare, usa un Waiter.

Ad esempio, questo frammento di codice crea una nuova tabella («myTable») in DynamoDB, attende che la tabella abbia uno ACTIVE stato, quindi stampa la risposta:

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

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

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

Per ulteriori informazioni, consulta [Uso dei camerieri](#).

## Client HTTP

È possibile modificare la configurazione predefinita per i client HTTP nelle applicazioni create con AWS SDK for Java. Per informazioni su come configurare i client e le impostazioni HTTP, consulta [Configurazione HTTP](#).

## Tentativi

È possibile modificare le impostazioni predefinite per i nuovi tentativi nei client di servizio, inclusa la modalità di ripetizione e la strategia di back-off. Per ulteriori informazioni, fate riferimento alla [RetryPolicy classe nell'API Reference](#). AWS SDK for Java

Per ulteriori informazioni sui nuovi tentativi nei AWS servizi, vedete [Ritentativi di errore e backoff esponenziale](#) in. AWS

## Timeout

È possibile configurare i timeout per ciascuno dei client di servizio utilizzando e i setter di.

[apiCallTimeoutapiCallAttemptTimeoutClientOverrideConfiguration.Builder](#)

L'apiCallTimeout impostazione indica la quantità di tempo necessaria per consentire al client di completare l'esecuzione di una chiamata API. L'apiCallAttemptTimeout impostazione indica il tempo di attesa per il completamento di ogni richiesta HTTP (nuovo tentativo) prima di rinunciare.

L'esempio seguente imposta entrambi i timeout per un client S3.

```
S3Client s3Client = S3Client.builder()
    .overrideConfiguration(b -> b
        .apiCallTimeout(Duration.ofSeconds(105L))
        .apiCallAttemptTimeout(Duration.ofSeconds(25L)))
    .build();
```

Puoi anche impostare i timeout a livello di richiesta configurando un oggetto di richiesta [AwsRequestOverrideConfiguration](#) fornendolo all'oggetto della richiesta con il metodo. `overrideConfiguration`

L'esempio seguente utilizza le stesse impostazioni di timeout ma a livello di richiesta per un'operazione S3. `PutObject`

```
S3Client basicS3Client = S3Client.create(); // Client with no timeout settings.

AwsRequestOverrideConfiguration overrideConfiguration =
    AwsRequestOverrideConfiguration.builder()
        .apiCallTimeout(Duration.ofSeconds(105L))
        .apiCallAttemptTimeout(Duration.ofSeconds(25L))
        .build();
```

```
basicS3Client.putObject(b -> b
    .bucket("DOC-EXAMPLE-BUCKET")
    .key("DOC-EXAMPLE_KEY")
    .overrideConfiguration(overrideConfiguration),
    RequestBody.fromString("test"));
```

## Intercettori di esecuzione

È possibile scrivere codice che intercetta l'esecuzione delle richieste e delle risposte API in diverse parti del ciclo di vita della richiesta/risposta. Ciò consente di pubblicare metriche, modificare una richiesta in corso, eseguire il debug dell'elaborazione delle richieste, visualizzare le eccezioni e altro ancora. Per ulteriori informazioni, consulta [l'ExecutionInterceptorinterfaccia](#) nell'API Reference.

AWS SDK for Java

## Informazioni aggiuntive

- Per esempi completi dei frammenti di codice precedenti, consulta [Lavorare con Amazon DynamoDB](#), [Lavorare con Amazon EC2](#) e [Lavorare con Amazon S3](#)

## Fornisci credenziali temporanee all'SDK

Prima di effettuare una richiesta di Amazon Web Services utilizzo di AWS SDK for Java 2.x, l'SDK firma crittograficamente le credenziali temporanee emesse da AWS. Per accedere alle credenziali temporanee, l'SDK recupera i valori di configurazione controllando diverse posizioni.

In questo argomento vengono descritti diversi modi per abilitare l'SDK per accedere alle credenziali temporanee.

### Argomenti

- [Configura l'accesso alle credenziali temporanee](#)
- [Catena di fornitori di credenziali predefinita](#)
- [Utilizza un fornitore o una catena di fornitori di credenziali specifico](#)
- [Usa i profili](#)
- [Carica credenziali temporanee da un processo esterno](#)



- [Fornisci credenziali temporanee nel codice](#)
- [Leggi le credenziali del ruolo IAM su Amazon EC2](#)

## Configura l'accesso alle credenziali temporanee

Per una maggiore sicurezza, si AWS consiglia di configurare l'SDK for Java in modo [da utilizzare credenziali temporanee](#) anziché credenziali di lunga durata. Le credenziali temporanee sono costituite da chiavi di accesso (ID della chiave di accesso e chiave di accesso segreta) e da un token di sessione. Ti consigliamo di [configurare l'SDK](#) per ottenere automaticamente le credenziali temporanee, poiché il processo di aggiornamento del token è automatico. Tuttavia, puoi [fornire direttamente all'SDK](#) credenziali temporanee.

### Configurazione di IAM Identity Center

Quando configuri l'SDK per utilizzare l'accesso Single Sign-On di IAM Identity Center come descritto [???](#) in questa guida, l'SDK utilizza automaticamente credenziali temporanee.

L'SDK utilizza il token di accesso IAM Identity Center per accedere al ruolo IAM configurato con l'impostazione nel file. `sso_role_name` config L'SDK assume questo ruolo IAM e recupera le credenziali temporanee da utilizzare per le richieste. Servizio AWS

Per maggiori dettagli su come l'SDK ottiene le credenziali temporanee dalla configurazione, consulta la sezione [Understanding IAM Identity Center dell'SDK and Tools Reference](#) Guide. AWS

### Recupera dal portale di accesso AWS

In alternativa alla configurazione Single Sign-On di IAM Identity Center, puoi copiare e utilizzare le credenziali temporanee disponibili nel portale di accesso. AWS È possibile utilizzare le credenziali temporanee in un profilo o utilizzarle come valori per le proprietà di sistema e le variabili di ambiente.

Imposta un file di credenziali locale per credenziali temporanee

1. [Crea un file di credenziali condiviso](#)
2. Nel file delle credenziali, incolla il seguente testo segnaposto fino a incollare le credenziali temporanee funzionanti.

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



codice crea una catena `DynamoDbClient` che utilizza la catena di provider di credenziali di default per individuare e recuperare le impostazioni di configurazione predefinite.

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

## Ordine di recupero delle impostazioni delle credenziali

La catena di provider di credenziali predefinita di SDK for Java 2.x cerca la configurazione nell'ambiente utilizzando una sequenza predefinita.

### 1. Proprietà del sistema Java

- L'SDK utilizza la [SystemPropertyCredentialsProvider](#) classe per caricare credenziali temporanee dalle proprietà di sistema `aws.accessKeyId`, `aws.secretAccessKey`, e `aws.sessionToken` Java.

#### Note

Per informazioni su come impostare le proprietà del sistema Java, consultate il tutorial [System Properties](#) sul sito Web ufficiale di Java Tutorials.

### 2. Variabili di ambiente

- L'SDK utilizza la [EnvironmentVariableCredentialsProvider](#) classe per caricare credenziali temporanee dalle variabili di ambiente `AWS_ACCESS_KEY_ID`, `AWS_SECRET_ACCESS_KEY`, e `AWS_SESSION_TOKEN`.

### 3. Token di identità Web da AWS Security Token Service

- L'SDK utilizza la [WebIdentityTokenFileCredentialsProvider](#) classe per caricare credenziali temporanee dalle proprietà del sistema Java o dalle variabili di ambiente.

### 4. I file condivisi `credentials` e `config`

- L'SDK li utilizza [ProfileCredentialsProvider](#) per caricare le impostazioni Single Sign-on o le credenziali temporanee di IAM Identity Center dal [default] profilo nei file condivisi e nei file `credentials config`.

La guida di riferimento agli AWS SDK e agli strumenti contiene [informazioni dettagliate](#) su come l'SDK for Java funziona con il token single sign-on IAM Identity Center per ottenere le credenziali temporanee che l'SDK utilizza per chiamare. Servizi AWS

#### Note

I config file `credentials` and sono condivisi da vari SDK e strumenti. AWS Per ulteriori informazioni, consulta [I file `.aws/credentials` e `.aws/config` nella Guida di riferimento agli SDK e agli strumenti.](#) AWS

#### 5. Amazon ECS credenziali del contenitore

- L'SDK utilizza la [ContainerCredentialsProvider](#) classe per caricare credenziali temporanee dalla variabile di ambiente di `AWS_CONTAINER_CREDENTIALS_RELATIVE_URI` sistema.

#### 6. Amazon EC2 ad esempio, credenziali fornite dal ruolo IAM

- L'SDK utilizza la [InstanceProfileCredentialsProvider](#) classe per caricare credenziali temporanee dal servizio di metadati. Amazon EC2

## Utilizza un fornitore o una catena di fornitori di credenziali specifico

In alternativa alla catena di provider di credenziali predefinita, puoi specificare quale provider di credenziali deve utilizzare l'SDK. Quando fornite un provider di credenziali specifico, l'SDK salta il processo di verifica delle varie ubicazioni, il che riduce leggermente il tempo necessario per creare un client di servizio.

Ad esempio, se imposti la configurazione predefinita utilizzando variabili di ambiente, fornisci un [EnvironmentVariableCredentialsProvider](#) oggetto al `credentialsProvider` metodo sul service client builder, come nel seguente frammento di codice.

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

Per un elenco completo dei fornitori di credenziali e delle catene di provider, consulta All Known Implementation Classes in. [AwsCredentialsProvider](#)

**Note**

È possibile utilizzare il proprio provider di credenziali o le proprie catene di fornitori implementando l'`AwsCredentialsProvider` interfaccia.

## Usa i profili

Utilizzando il `credentials` file condiviso `config` e, puoi configurare diversi profili. Ciò consente all'applicazione di utilizzare più set di configurazione delle credenziali. Il `[default]` profilo è stato menzionato in precedenza. L'SDK utilizza la [ProfileCredentialsProvider](#) classe per caricare le impostazioni dai profili definiti nel `credentials` file condiviso.

Il seguente frammento di codice mostra come creare un client di servizio che utilizzi le impostazioni definite come parte del profilo denominato `my_profile`

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

## Imposta un profilo diverso come predefinito

Per impostare un profilo diverso dal `[default]` profilo come predefinito per l'applicazione, imposta la variabile di `AWS_PROFILE` ambiente sul nome del profilo personalizzato.

Per impostare questa variabile su Linux, macOS o Unix, usa: `export`

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

Per impostare queste variabili su Windows, utilizza `set`:

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

In alternativa, impostate la proprietà del sistema `aws.profile` Java sul nome del profilo.

## Ricarica le credenziali del profilo

È possibile configurare qualsiasi provider di credenziali che disponga di un `profileFile()` metodo nel relativo generatore per ricaricare le credenziali del profilo. Queste classi di profilo di credenziali sono: `ProfileCredentialsProvider`, `DefaultCredentialsProvider`, `InstanceProfileCredentialsProvider` e `ProfileTokenProvider`.

### Note

Il ricaricamento delle credenziali del profilo funziona solo con le seguenti impostazioni nel file di profilo: `aws_access_key_id`, `aws_secret_access_key` e `aws_session_token`. Impostazioni come `region`, `sso_session_sso_account_id`, e `source_profile` vengono ignorate.

Per configurare un provider di credenziali supportato per ricaricare le impostazioni del profilo, fornisci un'istanza del metodo `ProfileFileSupplier` builder. `profileFile()` Il seguente esempio di codice mostra un `ProfileCredentialsProvider` che ricarica le impostazioni delle credenziali dal profilo. [default]

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

// Set up a service client with the provider instance.
DynamoDbClient dynamoDbClient = DynamoDbClient.builder()
    .region(Region.US_EAST_1)
    .credentialsProvider(provider)
    .build();

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

Quando `ProfileCredentialsProvider.resolveCredentials()` viene chiamato, l'SDK for Java ricarica le impostazioni. `ProfileFileSupplier.defaultSupplier()` è una delle [numeroso implementazioni pratiche](#) `ProfileFileSupplier` fornite dall'SDK. Se il tuo caso d'uso lo richiede, puoi fornire la tua implementazione.

L'esempio seguente mostra l'uso del metodo di

`ProfileFileSupplier.reloadWhenModified()` convenienza.

`reloadWhenModified()` utilizza un `Path` parametro, che offre flessibilità nella designazione del file sorgente per la configurazione anziché la posizione standard `~/.aws/credentials` (`oconfig`).

Le impostazioni verranno ricaricate quando `resolveCredentials()` viene chiamata solo se SDK determina che il contenuto del file è stato modificato.

```
Path credentialsFilePath = ...

ProfileCredentialsProvider provider = ProfileCredentialsProvider
    .builder()
    .profileFile(ProfileFileSupplier.reloadWhenModified(credentialsFilePath,
ProfileFile.Type.CREDENTIALS))
    .profileName("my-profile")
    .build();
/*
   A service client configured with the provider instance calls
   provider.resolveCredential()
   before each request.
*/
```

Il `ProfileFileSupplier.aggregate()` metodo unisce il contenuto di più file di configurazione. Siete voi a decidere se ricaricare un file per chiamata `resolveCredentials()` o se le impostazioni di un file vengono corrette al momento della prima lettura.

L'esempio seguente mostra un file `DefaultCredentialsProvider` che unisce le impostazioni di due file che contengono le impostazioni del profilo. L'SDK ricarica le impostazioni nel file a cui fa riferimento la `credentialsFilePath` variabile ogni volta che `resolveCredentials()` viene chiamata e le impostazioni vengono modificate. Le impostazioni dell'`profileFile` oggetto rimangono le stesse.

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

DefaultCredentialsProvider provider = DefaultCredentialsProvider
    .builder()
    .profileFile(ProfileFileSupplier.aggregate(
        ProfileFileSupplier.reloadWhenModified(credentialsFilePath,
ProfileFile.Type.CREDENTIALS),
        ProfileFileSupplier.fixedProfileFile(profileFile)))
```

```
        .profileName("my-profile")
        .build();
/*
   A service client configured with the provider instance calls
   provider.resolveCredential()
   before each request.
*/
```

## Carica credenziali temporanee da un processo esterno

### Warning

Di seguito viene descritto un metodo per ottenere credenziali temporanee da un processo esterno. Questo può essere potenzialmente pericoloso, quindi procedi con cautela. Se possibile, dovrebbero essere preferiti altri fornitori di credenziali. Se si utilizza questa opzione, è necessario assicurarsi che il `config` file sia il più protetto possibile utilizzando le migliori pratiche di sicurezza per il sistema operativo in uso.

Assicurati che lo strumento per le credenziali personalizzate non scriva `StdErr` informazioni segrete su. Gli SDK AWS CLI possono acquisire e registrare tali informazioni, esponendole potenzialmente a utenti non autorizzati.

Con l'SDK for Java 2.x, puoi acquisire credenziali temporanee da un processo esterno per casi d'uso personalizzati. Esistono due modi per configurare questa funzionalità.

### Usa l'`credential_process` impostazione

Se disponi di un metodo che fornisce credenziali temporanee, puoi integrarlo aggiungendo l'`credential_process` impostazione come parte di una definizione di profilo nel `config` file. Il valore specificato deve utilizzare il percorso completo del file di comando. Se il percorso del file contiene spazi, è necessario racchiuderlo tra virgolette.

L'SDK chiama il comando esattamente come indicato e quindi legge i dati JSON da `stdout`

Gli esempi seguenti mostrano l'uso di questa impostazione per percorsi di file senza spazi e percorsi di file con spazi.



## Linux/macOS

### Nessuno spazio nel percorso del file

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

### Spazi nel percorso del file

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

## Windows

### Nessuno spazio nel percorso del file

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

### Spazi nel percorso del file

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

Il seguente frammento di codice mostra come creare un client di servizio che utilizza le credenziali temporanee definite come parte del profilo denominato `process-credential-profile`

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

Per informazioni dettagliate sull'utilizzo di un processo esterno come fonte di credenziali temporanee, consulta la [sezione relativa alle credenziali di processo nella Guida di riferimento agli SDK](#) e agli AWS strumenti.

## Utilizzo dell'**ProcessCredentialsProvider**

In alternativa all'utilizzo delle impostazioni nel config file, puoi utilizzare gli SDK [ProcessCredentialsProvider](#) per caricare credenziali temporanee utilizzando Java.

Gli esempi seguenti mostrano varie versioni di come specificare un processo esterno utilizzando `ProcessCredentialsProvider` e configurare un client di servizio che utilizza le credenziali temporanee.

### Linux/macOS

#### Nessuno spazio nel percorso del file

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

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

#### Spazi nel percorso del file

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

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

## Windows

### Nessuno spazio nel percorso del file

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

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

### Spazi nel percorso del file

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

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

## Fornisci credenziali temporanee nel codice

Se la catena di credenziali predefinita o un provider o una catena di fornitori specifici o personalizzati non funzionano per l'applicazione, puoi fornire credenziali temporanee direttamente nel codice. Queste possono essere [credenziali di ruolo IAM](#) come [descritto sopra](#) o credenziali temporanee recuperate da `AWSSecurityTokenServiceClient`. AWS Security Token Service (AWS STS) Se hai recuperato credenziali temporanee utilizzando AWS STS, forniscile a un Servizio AWS client come mostrato nel seguente esempio di codice.

1. Assumi un ruolo chiamando `StsClient.assumeRole()`
2. Crea un [StaticCredentialsProvider](#) oggetto e forniscilo con l'`AwsSessionCredentials` oggetto.

### 3. Configura il service client builder con `StaticCredentialsProvider` e crea il client.

L'esempio seguente crea un client di servizio Amazon S3 utilizzando credenziali temporanee restituite da AWS STS per un ruolo assunto da IAM.

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

    // The SDK's default credentials provider chain will find the single sign-on
    // settings in the [default] profile.
    // The identity configured with the [default] profile needs permission to call
    // AssumeRole on the STS service.
    try {
        Credentials tempRoleCredentials;
        try (StsClient stsClient = StsClient.create()) {
            AssumeRoleRequest roleRequest = AssumeRoleRequest.builder()
                .roleArn(roleArn)
                .roleSessionName(roleSessionName)
                .build();

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

        // List all buckets in the account associated with the assumed role
        // by using the temporary credentials retrieved by invoking
        stsClient.assumeRole().
        StaticCredentialsProvider staticCredentialsProvider =
        StaticCredentialsProvider.create(
            AwsSessionCredentials.create(key, secKey, secToken));
        try (S3Client s3 = S3Client.builder()
            .credentialsProvider(staticCredentialsProvider)
            .build()) {
            List<Bucket> buckets = s3.listBuckets().buckets();
        }
    }
}
```

```
        for (Bucket bucket : buckets) {
            System.out.println("bucket name: " + bucket.name());
        }
    }
} catch (StsException | S3Exception e) {
    logger.error(e.getMessage());
    System.exit(1);
}
}
```

## Set di autorizzazioni

Il seguente set di autorizzazioni definito in AWS IAM Identity Center consente all'identità (utente) di eseguire le due operazioni seguenti

1. Il `GetObject` funzionamento di Amazon Simple Storage Service.
2. Il `AssumeRole` funzionamento di AWS Security Token Service.

Senza assumere il ruolo, il `s3.listBuckets()` metodo illustrato nell'esempio fallirebbe.

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

## Ruolo presunto

### Politica sulle autorizzazioni relative ai ruoli assunti

La seguente politica di autorizzazioni è associata al ruolo assunto nell'esempio precedente. Questa politica di autorizzazioni consente di elencare tutti i bucket nello stesso account del ruolo.

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

## Politica di fiducia per i ruoli assunti

La seguente politica di fiducia è associata al ruolo assunto nell'esempio precedente. La politica consente alle identità (utenti) di assumere il ruolo in due account.

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

## Leggi le credenziali del ruolo IAM su Amazon EC2

Puoi utilizzare un ruolo IAM per gestire le credenziali temporanee per le applicazioni in esecuzione su un'istanza EC2 e che AWS CLI effettuano richieste API. AWS Ciò è preferibile all'archiviazione

delle chiavi di accesso nell'istanza EC2. Per assegnare un AWS ruolo a un'istanza EC2 e renderlo disponibile per tutte le sue applicazioni, crei un profilo di istanza collegato all'istanza. Un profilo dell'istanza contiene il ruolo e consente ai programmi in esecuzione sull'istanza EC2 di ottenere le credenziali temporanee. Per ulteriori informazioni, consulta [Utilizzo di un ruolo IAM per concedere autorizzazioni ad applicazioni in esecuzione su istanze di Amazon EC2](#) nella Guida per l'utente di IAM.

Questo argomento fornisce informazioni su come configurare l'applicazione Java per l'esecuzione su un'istanza EC2 e abilitare l'SDK for Java per IAM acquisire le credenziali di ruolo.

## Acquisisci le credenziali del ruolo IAM dall'ambiente

Se l'applicazione crea un client di AWS servizio utilizzando il `create` metodo (o i `builder().build()` metodi), l'SDK for Java utilizza la catena di provider di credenziali predefinita. La catena di fornitori di credenziali predefinita cerca nell'ambiente di esecuzione gli elementi di configurazione che l'SDK può scambiare con credenziali temporanee. La [the section called “Catena di fornitori di credenziali predefinita”](#) sezione descrive il processo di ricerca completo.

L'ultimo passaggio della catena di provider predefinita è disponibile solo quando l'applicazione viene eseguita su un' Amazon EC2 istanza. In questa fase, l'SDK utilizza un `InstanceProfileCredentialsProvider` per leggere il ruolo IAM definito nel profilo dell'istanza EC2. L'SDK acquisisce quindi le credenziali temporanee per quel ruolo IAM.

Sebbene queste credenziali siano temporanee e alla fine scadano, An le aggiorna `InstanceProfileCredentialsProvider` periodicamente in modo che continuino a consentirne l'accesso. AWS

## Acquisisci le credenziali del ruolo IAM a livello di codice

In alternativa alla catena di provider di credenziali predefinita che alla fine utilizza un client di servizio `InstanceProfileCredentialsProvider` su EC2, puoi configurare un client di servizio in modo esplicito con un `InstanceProfileCredentialsProvider` Questo approccio è illustrato nel frammento seguente.

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

## Acquisisci in modo sicuro le credenziali dei ruoli IAM

Per impostazione predefinita, le istanze EC2 eseguono [IMDS](#) (Instance Metadata Service) che consente agli SDK di accedere `InstanceProfileCredentialsProvider` a informazioni come il ruolo IAM che è stato configurato. Per impostazione predefinita, le istanze EC2 eseguono due versioni di IMDS:

- Servizio di metadati dell'istanza Versione 1 (IMDSv1): un metodo di richiesta/risposta
- Servizio di metadati dell'istanza Versione 2 (IMDSv2): un metodo orientato alla sessione

### [IMDSv2 è un approccio più sicuro di IMDSv1.](#)

Per impostazione predefinita, Java SDK prova innanzitutto IMDSv2 per ottenere il ruolo IAM, ma se ciò fallisce, prova IMDSv1. Tuttavia, poiché IMDSv1 è meno sicuro, AWS consiglia di utilizzare solo IMDSv2 e di impedire all'SDK di provare IMDSv1.

Per utilizzare un approccio più sicuro, disabilita l'SDK dall'utilizzo di IMDSv1 fornendo una delle seguenti impostazioni con un valore di `true`

- Variabile d'ambiente: `AWS_EC2_METADATA_V1_DISABLED`
- Proprietà del sistema JVM: `aws.disableEc2MetadataV1`
- Impostazione del file di configurazione condiviso: `ec2_metadata_v1_disabled`

Con una di queste impostazioni impostata su `true`, l'SDK non carica le credenziali del ruolo IMDS utilizzando IMDSv1 se la chiamata IMDSv2 iniziale fallisce.

## Usa Regioni AWS

Regioni AWS consentire l'accesso ai client del servizio Servizi AWS che risiedono fisicamente in un'area geografica specifica.

### configura esplicitamente un Regione AWS

[Per impostare in modo esplicito una regione, si consiglia di utilizzare le costanti definite nella classe `Region`.](#) Si tratta di un'enumerazione di tutte le regioni disponibili pubblicamente.

Per creare un client con una regione enumerata dalla classe, utilizzate il metodo del client builder. `region`



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

Se la regione che desideri utilizzare non è una delle enumerazioni della `Region` classe, puoi creare una nuova regione utilizzando il metodo statico `of`. Questo metodo consente di accedere a nuove regioni senza aggiornare l'SDK.

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

### Note

Dopo aver creato un client con il builder, è immutabile e non può essere modificato. Regione AWS Se devi lavorare con più client Regioni AWS per lo stesso servizio, devi creare più client, uno per regione.

## Consenti all'SDK di determinare automaticamente la regione dall'ambiente

Quando il codice viene eseguito su Amazon EC2 o AWS Lambda, potresti voler configurare i client in modo Regione AWS che utilizzino lo stesso su cui è in esecuzione il codice. Questo disaccoppia il codice dall'ambiente in cui è in esecuzione e semplifica la distribuzione dell'applicazione su più utenti Regioni AWS per ridurre la latenza o la ridondanza.

Per utilizzare la catena di provider di credenziali/regioni predefinita per determinare la regione dall'ambiente, utilizzate il metodo del client builder `create`

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

Se non impostate una regione in modo esplicito Regione AWS utilizzando il `region` metodo, l'SDK consulta la catena di provider di regioni predefinita per determinare la regione da utilizzare.

## Comprendi la catena di fornitori regionali predefinita

L'SDK esegue le seguenti operazioni per cercare un Regione AWS :

1. Qualsiasi regione esplicita impostata utilizzando il builder stesso ha la precedenza `region` su qualsiasi altra cosa.
2. La variabile di ambiente `AWS_REGION` è selezionata. Se è impostata, quella regione viene utilizzata per configurare il client.

#### Note

Il Lambda contenitore imposta questa variabile di ambiente.

3. L'SDK controlla il file di configurazione AWS condiviso e il file di credenziali condivise (di solito si trova in `~/.aws/config` and `~/.aws/credentials`). Se la `region` proprietà è presente, l'SDK la utilizza.
  - Se l'SDK trova la `region` proprietà in entrambi i file per lo stesso profilo (incluso il `default` profilo), l'SDK utilizza il valore nel file delle credenziali condivise.
  - La variabile di ambiente `AWS_CONFIG_FILE` può essere utilizzata per personalizzare il percorso del file di configurazione condiviso.
  - La variabile di `AWS_PROFILE` ambiente o la proprietà di `aws.profile` sistema possono essere utilizzate per specificare il profilo caricato dall'SDK.
4. L'SDK tenta di utilizzare il servizio di metadati dell' Amazon EC2 istanza (IMDS) per determinare la regione dell'istanza attualmente in esecuzione. Amazon EC2
  - Per una maggiore sicurezza, dovresti disattivare l'SDK dal tentativo di utilizzare la versione 1 di IMDS. Per disabilitare la versione 1, si utilizza la stessa impostazione descritta nella sezione. [the section called “In modo sicuro”](#)
5. Se l'SDK non ha ancora trovato una regione a questo punto, la creazione del client fallisce con un'eccezione.

Durante lo sviluppo di AWS applicazioni, un approccio comune consiste nell'utilizzare il file di configurazione condiviso (descritto nell'[ordine di recupero delle credenziali](#)) per impostare la regione per lo sviluppo locale e fare affidamento sulla catena di provider di regioni predefinita per determinare la regione in cui l'applicazione viene eseguita sull'infrastruttura. AWS Questo semplifica notevolmente la creazione del client e mantiene l'applicazione portatile.

## Verifica la disponibilità del servizio in una regione

Per verificare se un determinato Servizio AWS prodotto è disponibile in una regione, utilizza il `region` metodo `serviceMetadata` and sul client del servizio.

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

Consultate la documentazione [della classe Region](#) per quanto riguarda la possibilità di Regioni AWS specificare e utilizzate il prefisso endpoint del servizio per eseguire le query.

## Scegli un endpoint specifico

In determinate situazioni, ad esempio per testare in anteprima le funzionalità di un servizio prima che queste diventino disponibili sul mercato, potrebbe essere necessario specificare un endpoint specifico in una regione. In queste situazioni, i client del servizio possono essere configurati chiamando il metodo `endpointOverride`

Ad esempio, per configurare un Amazon EC2 client in modo che utilizzi la regione Europa (Irlanda) con un endpoint specifico, utilizza il codice seguente.

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

Vedi [Regioni ed endpoint](#) per l'elenco corrente delle regioni e gli endpoint corrispondenti per tutti i servizi. AWS

## Riduci i tempi di avvio dell'SDK per AWS Lambda

Uno degli obiettivi di AWS SDK for Java 2.x è ridurre la latenza di avvio delle funzioni AWS Lambda. L'SDK contiene modifiche che riducono i tempi di avvio, discusse alla fine di questo argomento.

Innanzitutto, questo argomento si concentra sulle modifiche che è possibile apportare per ridurre i tempi di avvio a freddo. Queste includono le modifiche alla struttura del codice e alla configurazione dei client di servizio.

## Utilizzate un client AWS HTTP basato su CRT

Per lavorare con AWS Lambda, consigliamo il [AwsCrtHttpClient](#) per scenari sincroni e il [AwsCrtAsyncHttpClient](#) per scenari asincroni.

L'[the section called "Configurazione AWS dei client HTTP basati su CRT"](#) argomento di questa guida descrive i vantaggi dell'utilizzo dei client HTTP, come aggiungere la dipendenza e come configurarne l'utilizzo da parte dei client di servizio.

## Rimuovi le dipendenze inutilizzate del client HTTP

Oltre all'uso esplicito di un client AWS basato su CRT, puoi rimuovere altri client HTTP che l'SDK integra per impostazione predefinita. Il tempo di avvio di Lambda si riduce quando è necessario caricare meno librerie, quindi è necessario rimuovere tutti gli artefatti inutilizzati che la JVM deve caricare.

Il seguente frammento di un `pom.xml` file Maven mostra l'esclusione del client HTTP basato su Apache e del client HTTP basato su Netty. (Questi client non sono necessari quando si utilizza un client basato su CRT). AWS Questo esempio esclude gli artefatti del client HTTP dalla dipendenza del client S3 e aggiunge l'`aws-crt-client` artefatto per consentire l'accesso ai client HTTP basati su CRT. AWS

```
<project>
  <properties>
    <aws.java.sdk.version>2.25.51</aws.java.sdk.version>
  </properties>
  <dependencyManagement>
    <dependencies>
      <dependency>
        <groupId>software.amazon.awssdk</groupId>
        <artifactId>bom</artifactId>
        <version>${aws.java.sdk.version}</version>
        <type>pom</type>
        <scope>import</scope>
      </dependency>
    </dependencies>
  </dependencyManagement>
  <dependencies>
    <dependency>
      <groupId>software.amazon.awssdk</groupId>
      <artifactId>aws-crt-client</artifactId>
    </dependency>
    <dependency>
      <groupId>software.amazon.awssdk</groupId>
      <artifactId>s3</artifactId>
      <exclusions>
        <exclusion>
          <groupId>software.amazon.awssdk</groupId>
          <artifactId>netty-nio-client</artifactId>
        </exclusion>
        <exclusion>
```

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

### Note

Aggiungi l'<exclusions>elemento a tutte le dipendenze del client di servizio nel tuo file. `pom.xml`

## Configura i client di servizio per abbreviare le ricerche

### Specificare una regione

Quando crei un client di servizio, chiama il `region` metodo sul service client builder. Questo accorcia il [processo di ricerca regionale predefinito dell'SDK, che controlla le](#) informazioni in diversi punti. Regione AWS

Per mantenere il codice Lambda indipendente dalla regione, usa il codice seguente all'interno del `region` metodo. Questo codice accede alla variabile di `AWS_REGION` ambiente impostata dal contenitore Lambda.

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

### Utilizzo dell'**EnvironmentVariableCredentialProvider**

Analogamente al comportamento di ricerca predefinito per le informazioni sulla regione, l'SDK cerca le credenziali in diversi punti. Specificando [EnvironmentVariableCredentialProvider](#) quando si crea un client di servizio, si risparmia tempo nel processo di ricerca dell'SDK.

### Note

L'utilizzo di questo provider di credenziali consente l'utilizzo del codice nelle Lambda funzioni, ma potrebbe non funzionare su Amazon EC2 altri sistemi.

Il seguente frammento di codice mostra un client di servizio S3 configurato in modo appropriato per l'uso in un ambiente Lambda.

```
S3Client s3Client = S3Client.builder()

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

## Inizializza il client SDK all'esterno del gestore di funzioni Lambda

Consigliamo di inizializzare un client SDK al di fuori del metodo del gestore Lambda. In questo modo, se il contesto di esecuzione viene riutilizzato, l'inizializzazione del client di servizio può essere ignorata. Riutilizzando l'istanza del client e le relative connessioni, le successive invocazioni del metodo handler avvengono più rapidamente.

Nell'esempio seguente, l'`S3Client` istanza viene inizializzata nel costruttore utilizzando un metodo factory statico. Se il contenitore gestito dall'ambiente Lambda viene riutilizzato, viene riutilizzata l'istanza inizializzata `S3Client`.

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

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

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

## Riduci al minimo l'iniezione di dipendenza

I framework di Dependency Injection (DI) potrebbero richiedere più tempo per completare il processo di configurazione. Potrebbero inoltre richiedere dipendenze aggiuntive, il cui caricamento richiede tempo.

[Se è necessario un framework DI, consigliamo di utilizzare framework DI leggeri come Dagger.](#)

## Usa un targeting Maven Archetype AWS Lambda

Il team AWS Java SDK ha sviluppato un [modello Maven Archetype](#) per avviare un progetto Lambda con tempi di avvio minimi. Puoi creare un progetto Maven a partire dall'archetipo e sapere che le dipendenze sono configurate in modo appropriato per l'ambiente Lambda.

[Per saperne di più sull'archetipo e illustrare un esempio di implementazione, consulta questo post sul blog.](#)

## Prendi in considerazione Lambda SnapStart per Java

Se i requisiti di runtime sono compatibili, AWS offre [Lambda SnapStart per Java](#). Lambda SnapStart è una soluzione basata sull'infrastruttura che migliora le prestazioni di avvio per le funzioni Java. Quando pubblichi una nuova versione di una funzione, Lambda la SnapStart inizializza e scatta un'istantanea crittografata e immutabile della memoria e dello stato del disco. SnapStart quindi memorizza l'istantanea nella cache per riutilizzarla.

## Modifiche alla versione 2.x che influiscono sul tempo di avvio

Oltre alle modifiche apportate al codice, la versione 2.x dell'SDK for Java include tre modifiche principali che riducono i tempi di avvio:

- Uso di [jackson-jr](#), una libreria di serializzazione che migliora i tempi di inizializzazione
- Uso delle librerie [java.time](#) per oggetti di data e ora, che fanno parte del JDK
- Uso di [SLF4j](#) per una facciata adibita a registrazione

## Risorse aggiuntive

La Guida per gli AWS Lambda sviluppatori contiene una [sezione sulle migliori pratiche](#) per lo sviluppo di funzioni Lambda che non è specifica per Java.

[Per un esempio di creazione di un'applicazione nativa per il cloud in Java che utilizza AWS Lambda, consulta il contenuto di questo workshop.](#) Il workshop discute dell'ottimizzazione delle prestazioni e di altre best practice.

Puoi prendere in considerazione l'utilizzo di immagini statiche compilate in anticipo per ridurre la latenza di avvio. Ad esempio, puoi utilizzare l'SDK per Java 2.x e Maven per [creare](#) un'immagine nativa GraalVM.

## Client HTTP

È possibile modificare il client HTTP da utilizzare per il client di servizio e modificare la configurazione predefinita per i client HTTP con AWS SDK for Java 2.x. Questa sezione illustra i client e le impostazioni HTTP per l'SDK.

### Client HTTP disponibili nell'SDK for Java

#### Client sincroni

I client HTTP sincroni nell'SDK for Java implementano [SdkHttpClient](#) l'interfaccia. Un client di servizio sincrono, come il `S3Client` o il `DynamoDbClient`, richiede l'uso di un client HTTP sincrono. AWS SDK for Java Offre tre client HTTP sincroni.

`ApacheHttpClient` (impostazione predefinita)

[ApacheHttpClient](#) è il client HTTP predefinito per i client di servizi sincroni. Per informazioni sulla configurazione di `ApacheHttpClient`, vedere [Configurare il client HTTP basato su Apache](#)

`AwsCrtHttpClient`

[AwsCrtHttpClient](#) fornisce un throughput elevato e un IO non bloccante. È basato sul client `Http` AWS Common Runtime (CRT). Per informazioni sulla configurazione `AwsCrtHttpClient` e sul suo utilizzo con i client di servizio, vedere [the section called “Configurazione AWS dei client HTTP basati su CRT”](#)

`URLConnectionHttpClient`

Per ridurre al minimo il numero di jar e librerie di terze parti utilizzate dall'applicazione, è possibile utilizzare il [URLConnectionHttpClient](#). Per informazioni sulla configurazione di `URLConnectionHttpClient`, vedere [Configurare il client HTTP basato su URLConnection](#)

#### Client asincroni

I client HTTP asincroni nell'SDK for Java implementano l'interfaccia [SdkAsyncHttpClient](#). Un client di servizio asincrono, come il `S3AsyncClient` o il `DynamoDbAsyncClient`, richiede l'uso di un client HTTP asincrono. Offre AWS SDK for Java due client HTTP asincroni.



## NettyNioAsyncHttpClient (impostazione predefinita)

[NettyNioAsyncHttpClient](#) è il client HTTP predefinito utilizzato dai client asincroni. Per informazioni sulla configurazione di `NettyNioAsyncHttpClient`, vedere [the section called “Configurazione del client HTTP basato su Netty”](#)

## AwsCrtAsyncHttpClient

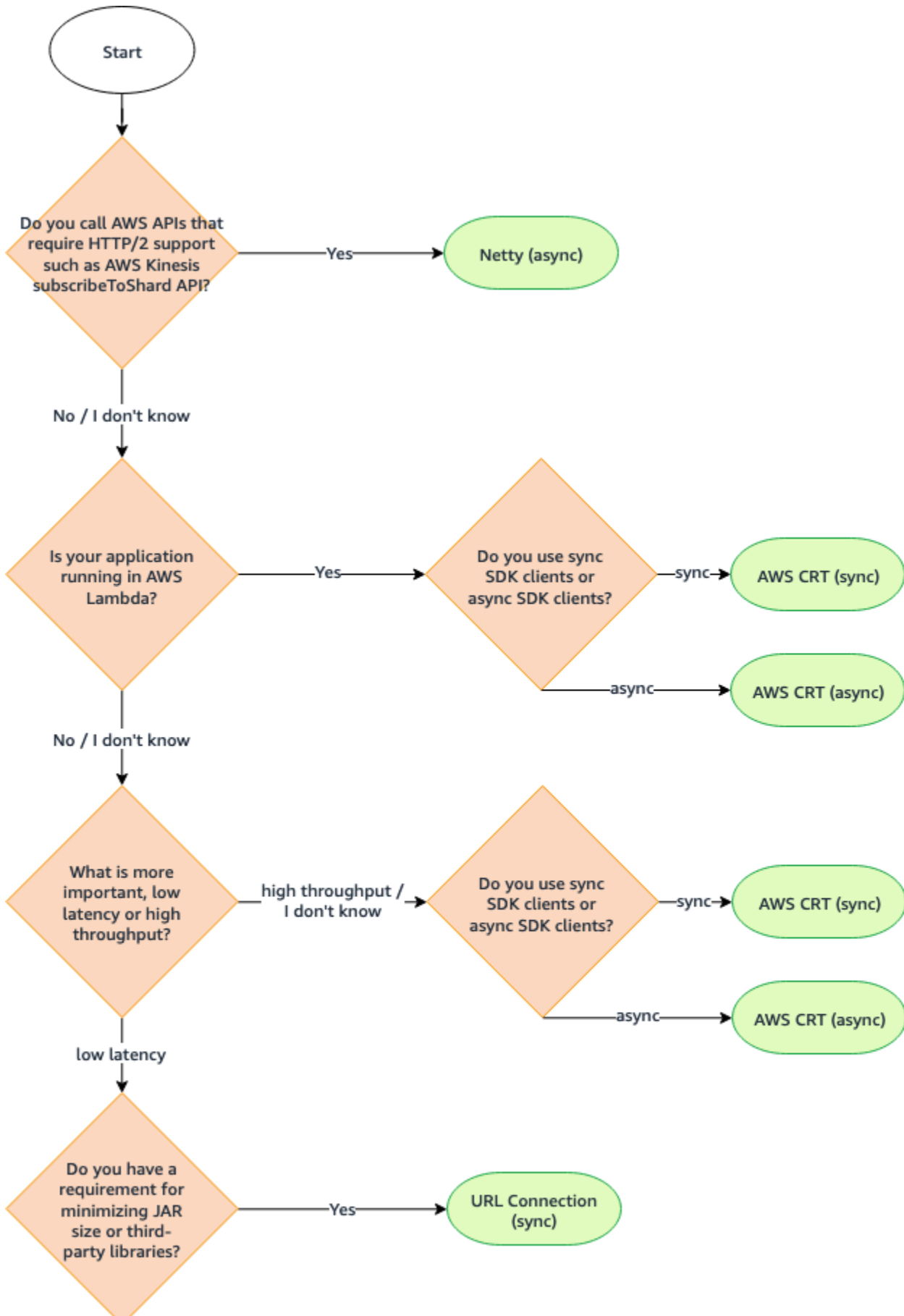
[AwsCrtAsyncHttpClient](#) si basa sul client HTTP AWS Common Runtime (CRT). Per informazioni sulla configurazione di `AwsCrtAsyncHttpClient`, vedere [the section called “Configurazione AWS dei client HTTP basati su CRT”](#)

## Raccomandazioni sui client HTTP

Quando si sceglie un'implementazione del client HTTP entrano in gioco diversi fattori. Utilizza le seguenti informazioni per aiutarti a decidere.

### diagramma di flusso dei consigli

Il seguente diagramma di flusso fornisce indicazioni generali per aiutarti a determinare quale client HTTP utilizzare.



## Confronto tra client HTTP

La tabella seguente fornisce informazioni dettagliate per ogni client HTTP.

Client HTTP	Sincronizzazione o asincrona	Quando usare	Limitazione/ inconveniente
Client HTTP basato su Apache  (client HTTP di sincronizzazione predefinito)	Sync	Usalo se preferisci una bassa latenza rispetto a un throughput elevato	Tempo di avvio più lento rispetto ad altri client HTTP
Client HTTP basato su URLConnection	Sync	Usalo se hai una forte esigenza di limitare le dipendenze di terze parti	Non supporta il metodo HTTP PATCH, richiesto da alcune API come le operazioni di Amazon APIGateway Update
AWS Client HTTP di sincronizzazione basato su CRT 1	Sync	<ul style="list-style-type: none"> <li>• Usalo se l'applicazione è in esecuzione in AWS Lambda</li> <li>• Usalo se preferisci un throughput elevato rispetto a una bassa latenza</li> <li>• Usalo se preferisci sincronizzare i client SDK</li> </ul>	N/D
Client HTTP basato su Netty	Asincrono	<ul style="list-style-type: none"> <li>• Usalo se la tua applicazione richiama API che richiedono il supporto HTTP/2 come l'API Kinesis <a href="#">SubscribeToShard</a></li> </ul>	Tempo di avvio più lento rispetto ad altri client HTTP

Client HTTP	Sincronizzazione o asincrona	Quando usare	Limitazione/ inconveniente
(client HTTP asincrono predefinito)			
AWS <sup>Client HTTP</sup> asincrono basato su CRT <sup>1</sup>	Asincrono	<ul style="list-style-type: none"> <li>• Usalo se l'applicazione è in esecuzione in AWS Lambda</li> <li>• Usalo se preferisci un throughput elevato rispetto a una bassa latenza</li> <li>• Usalo se preferisci i client SDK asincroni</li> </ul>	<ul style="list-style-type: none"> <li>• Non supporta client di servizio che richiedono o il supporto HTTP/2 come e KinesisAsyncClient TranscribeStreamingAsyncClient</li> </ul>

<sup>1</sup> A causa dei loro vantaggi aggiuntivi, si consiglia di utilizzare, se possibile, i client HTTP AWS basati su CRT.

## Impostazioni predefinite di configurazione intelligente

La AWS SDK for Java 2.x (versione 2.17.102 o successiva) offre una funzionalità di configurazione predefinita intelligente. Questa funzionalità ottimizza due proprietà del client HTTP insieme ad altre proprietà che non influiscono sul client HTTP.

Le impostazioni predefinite di configurazione intelligente impostano valori ragionevoli per le `tlsNegotiationTimeoutInMillis` proprietà `connectTimeoutInMillis` and in base a un valore della modalità `defaults` fornito dall'utente. Scegliete il valore della modalità predefinita in base alle caratteristiche dell'applicazione.

[Per ulteriori informazioni sulle impostazioni predefinite di configurazione intelligente e su come scegliere il valore della modalità predefinita più adatto alle applicazioni, consulta la Guida di riferimento agli SDK e agli AWS strumenti.](#)

Di seguito sono riportati quattro modi per impostare la modalità predefinita per l'applicazione.

## Service client

Utilizzate il service client builder per configurare la modalità predefinita direttamente sul client del servizio. L'esempio seguente imposta la modalità predefinita su. auto `DynamoDbClient`

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

## System property

È possibile utilizzare la proprietà `aws.defaultsMode` di sistema per specificare la modalità predefinita. Se si imposta la proprietà di sistema in Java, è necessario impostare la proprietà prima di inizializzare qualsiasi client di servizio.

L'esempio seguente mostra come impostare la modalità predefinita sull'autoutilizzo di una proprietà di sistema impostata in Java.

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

L'esempio seguente mostra come impostare la modalità di default per l'autoutilizzo di un'opzione del comando. `java`

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

## Environment variable

Imposta un valore per la variabile di ambiente `AWS_DEFAULTS_MODE` per selezionare la modalità predefinita per l'applicazione.

Le informazioni seguenti mostrano il comando da eseguire per impostare il valore della modalità predefinita sull'autoutilizzo di una variabile di ambiente.

Sistema operativo	Comando per impostare le variabili di ambiente
Linux, macOS o Unix	<code>export AWS_DEFAULTS_MODE=auto</code>
Windows	<code>set AWS_DEFAULTS_MODE=auto</code>

## AWS config file

È possibile aggiungere una proprietà di `defaults_mode` configurazione al AWS config file condiviso come illustrato nell'esempio seguente.

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

Se impostate la modalità predefinita a livello globale con la proprietà di sistema, la variabile di ambiente o il file di AWS configurazione, potete sovrascrivere le impostazioni quando create un client HTTP.

Quando create un client HTTP con il `httpClientBuilder()` metodo, le impostazioni si applicano solo all'istanza che state creando. Un esempio di ciò è mostrato [qui](#). Il client HTTP basato su Netty in questo esempio sostituisce tutti i valori di modalità predefiniti impostati globalmente per `and.connectTimeoutInMillis` `tlsNegotiationTimeoutInMillis`

## Configurare il client HTTP basato su Apache

Per impostazione predefinita, i client di servizio sincroni AWS SDK for Java 2.x utilizzano un client HTTP basato su Apache. [ApacheHttpClient](#) L'SDK è basato su `ApacheHttpClient` Apache. [HttpClient](#)

L'SDK offre anche il [URLConnectionHttpClient](#), che si carica più rapidamente, ma ha meno funzionalità. Per informazioni sulla configurazione di `URLConnectionHttpClient`, consulta [the section called “Configurare il client HTTP basato su URLConnection”](#)

[Per visualizzare il set completo di opzioni di configurazione disponibili perApacheHttpClient, consulta ApacheHttpClient.Builder e .Builder.ProxyConfiguration](#)

## Accedi a **ApacheHttpClient**

Nella maggior parte dei casi, si utilizza `ApacheHttpClient` senza alcuna configurazione esplicita. Dichiarare i tuoi client di servizio e l'SDK li configurerà automaticamente `ApacheHttpClient` con valori standard.

Se desideri configurarlo in modo esplicito `ApacheHttpClient` o utilizzarlo con più client di servizio, devi renderlo disponibile per la configurazione.

## Non è necessaria alcuna configurazione

Quando si dichiara una dipendenza da un client di servizio in Maven, l'SDK aggiunge una dipendenza di runtime dall'artefatto. `apache-client` Ciò rende la `ApacheHttpClient` classe disponibile per il codice in fase di esecuzione, ma non in fase di compilazione. Se non state configurando il client HTTP basato su Apache, non è necessario specificare una dipendenza per esso.

Nel seguente frammento XML di un `pom.xml` file Maven, la dipendenza dichiarata con `introduceAutomatically=true` introduce automaticamente il client HTTP basato su Apache. `<artifactId>s3</artifactId>` Non è necessario dichiarare una dipendenza specifica per questo.

```
<dependencyManagement>
  <dependencies>
    <dependency>
      <groupId>software.amazon.awssdk</groupId>
      <artifactId>bom</artifactId>
      <version>2.17.290</version>
      <type>pom</type>
      <scope>import</scope>
    </dependency>
  </dependencies>
</dependencyManagement>
<dependencies>
  <!-- The s3 dependency automatically adds a runtime dependency on the
  ApacheHttpClient-->
  <dependency>
    <groupId>software.amazon.awssdk</groupId>
    <artifactId>s3</artifactId>
  </dependency>
</dependencies>
```

Con queste dipendenze, non è possibile apportare modifiche esplicite alla configurazione HTTP, poiché la `ApacheHttpClient` libreria si trova solo nel classpath di runtime.

## Configurazione necessaria

Per configurare `ApacheHttpClient`, è necessario aggiungere una dipendenza dalla libreria in fase di compilazione `apache-client`.

Fate riferimento al seguente esempio di `pom.xml` file Maven per configurare `ApacheHttpClient`

```
<dependencyManagement>
```

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

## Usa e configura il **ApacheHttpClient**

È possibile configurare un'istanza `ApacheHttpClient` insieme alla creazione di un client di servizio oppure configurare una singola istanza da condividere tra più client di servizio.

Con entrambi gli approcci, si utilizza [ApacheHttpClient.Builder](#) per configurare le proprietà per il client HTTP basato su Apache.

Procedura consigliata: dedicare un'**ApacheHttpClient**istanza a un client di servizio

Se devi configurare un'istanza di `ApacheHttpClient`, ti consigliamo di creare l'`ApacheHttpClient`istanza dedicata. Puoi farlo utilizzando il `httpClientBuilder` metodo del builder del client del servizio. In questo modo, il ciclo di vita del client HTTP viene gestito dall'SDK, che aiuta a evitare potenziali perdite di memoria se l'`ApacheHttpClient`istanza non viene chiusa quando non è più necessaria.

L'esempio seguente crea `S3Client` e configura l'istanza incorporata di `withAndValues`. `ApacheHttpClient` `maxConnections` `connectionTimeout` L'istanza HTTP viene creata utilizzando il `httpClientBuilder` metodo di `S3Client.Builder`.



## Importazioni

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

## Codice

```
S3Client s3Client = S3Client // Singleton: Use the s3Client for all requests.  
    .builder()  
    .httpClientBuilder(ApacheHttpClient.builder()  
        .maxConnections(100)  
        .connectionTimeout(Duration.ofSeconds(5))  
    ).build();  
  
// Perform work with the s3Client.  
  
s3Client.close(); // Requests completed: Close all service clients.
```

## Approccio alternativo: condividere un'ApacheHttpClientistanza

Per ridurre l'utilizzo di risorse e memoria per l'applicazione, è possibile configurarne un'applicazione `ApacheHttpClient` e condividerla tra più client di servizio. Il pool di connessioni HTTP verrà condiviso, il che riduce l'utilizzo delle risorse.

### Note

Quando un'ApacheHttpClientistanza è condivisa, è necessario chiuderla quando è pronta per essere eliminata. L'SDK non chiuderà l'istanza quando il client del servizio viene chiuso.

L'esempio seguente configura un client HTTP basato su Apache utilizzato da due client di servizio. L'ApacheHttpClientistanza configurata viene passata al `httpClient` metodo di ogni builder. Quando i client di servizio e il client HTTP non sono più necessari, il codice li chiude esplicitamente. Il codice chiude per ultimo il client HTTP.

## Importazioni

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

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

## Codice

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

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

DynamoDbClient dynamoDbClient =
    DynamoDbClient.builder()
        .httpClient(apacheHttpClient).build();

// Perform work with the s3Client and dynamoDbClient.

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

## Esempio di configurazione del proxy

Il seguente frammento di codice utilizza il [generatore di configurazione proxy per il client HTTP Apache](#).

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

Le proprietà di sistema Java equivalenti per la configurazione del proxy sono mostrate nel seguente frammento di riga di comando.

```
$ java -Dhttp.proxyHost=example.com -Dhttp.proxyPort=1234 -Dhttp.proxyUser=username \
```

```
-Dhttp.proxyPassword=password -Dhttp.nonProxyHosts=localhost|host.example.com -cp ...  
App
```

La configurazione equivalente che utilizza le variabili di ambiente è:

```
// Set the following environment variables.  
// $ export HTTP_PROXY="http://username:password@example.com:1234"  
// $ export NO_PROXY="localhost|host.example.com"  
  
// Set the 'useSystemPropertyValues' to false on the proxy configuration.  
SdkHttpClient apacheHttpClient = ApacheHttpClient.builder()  
    .proxyConfiguration(ProxyConfiguration.builder()  
        .useSystemPropertyValues(Boolean.FALSE)  
        .build())  
    .build();  
  
// Run the application.  
// $ java -cp ... App
```

#### Note

Il client HTTP Apache attualmente non supporta le proprietà del sistema proxy HTTPS o la variabile di ambiente HTTPS\_PROXY.

## Configurare il client HTTP basato su URLConnection

AWS SDK for Java 2.x Offre un client [URLConnectionHttpClient](#) HTTP più leggero rispetto a quello predefinito. `URLConnectionHttpClient` è basato su Java.

### [URLConnection](#)

Si `URLConnectionHttpClient` carica più rapidamente rispetto al client HTTP basato su Apache, ma ha meno funzionalità. Poiché si carica più rapidamente, è una [buona soluzione](#) per le funzioni Java. AWS Lambda

`URLConnectionHttpClient` ha diverse [opzioni configurabili](#) a cui è possibile accedere.

#### Note

Non `URLConnectionHttpClient` supporta il metodo HTTP PATCH.

Alcune operazioni AWS API richiedono richieste PATCH. I nomi di queste operazioni di solito iniziano con `Update*`. Di seguito sono riportati alcuni esempi.

- [Diverse Update\\* operazioni](#) nell' AWS Security Hub API e anche l'[BatchUpdateFindings](#) operazione
- Tutte le [Update\\* operazioni](#) API di Amazon API Gateway
- [Diverse Update\\* operazioni](#) nell' WorkDocs API Amazon

Se puoi utilizzare il `URLConnectionHttpClient`, consulta innanzitutto l'API Reference relativo a Servizio AWS quello che stai utilizzando. Verifica se le operazioni necessarie utilizzano l'operazione PATCH.

## Accedi al `URLConnectionHttpClient`

Per configurare e utilizzare `URLConnectionHttpClient`, dichiara una dipendenza dall'artefatto `url-connection-client` Maven nel tuo file `pom.xml`

A differenza di `ApacheHttpClient`, non `URLConnectionHttpClient` viene aggiunto automaticamente al progetto, quindi use deve dichiararlo specificamente.

Il seguente esempio di `pom.xml` file mostra le dipendenze necessarie per utilizzare e configurare il client HTTP.

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

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

<dependencies>
  <dependency>
```

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

## Usa e configura il `URLConnectionHttpClient`

È possibile configurare un'istanza `URLConnectionHttpClient` insieme alla creazione di un client di servizio oppure configurare una singola istanza da condividere tra più client di servizio.

Con entrambi gli approcci, si utilizza [URLConnectionHttpClient.Builder](#) per configurare le proprietà per il client HTTP basato su `URLConnection`.

Procedura consigliata: dedicare un'istanza a un client di servizio **`URLConnectionHttpClient`**

Se devi configurare un'istanza di `URLConnectionHttpClient`, ti consigliamo di creare l'`URLConnectionHttpClient` istanza dedicata. Puoi farlo utilizzando il `httpClientBuilder` metodo del builder del client del servizio. In questo modo, il ciclo di vita del client HTTP viene gestito dall'SDK, che aiuta a evitare potenziali perdite di memoria se l'`URLConnectionHttpClient` istanza non viene chiusa quando non è più necessaria.

L'esempio seguente crea `S3Client` e configura l'istanza incorporata di `with and values`. `URLConnectionHttpClient` `socketTimeout` `proxyConfiguration`

Il `proxyConfiguration` metodo accetta un'espressione lambda Java di tipo.

`Consumer<ProxyConfiguration.Builder>`

### Importazioni

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

### Codice

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

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

// Perform work with the s3Client.

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

## Approccio alternativo: condividi un'istanza **URLConnectionHttpClient**

Per ridurre l'utilizzo di risorse e memoria per l'applicazione, è possibile configurarne un'applicazione `URLConnectionHttpClient` e condividerla tra più client di servizio. Il pool di connessioni HTTP verrà condiviso, il che riduce l'utilizzo delle risorse.

### Note

Quando un'`URLConnectionHttpClient` istanza è condivisa, è necessario chiuderla quando è pronta per essere eliminata. L'SDK non chiuderà l'istanza quando il client del servizio viene chiuso.

L'esempio seguente configura un client HTTP basato su `URLConnection` utilizzato da due client di servizio. L'`URLConnectionHttpClient` istanza configurata viene passata al `httpClient` metodo di ogni builder. Quando i client di servizio e il client HTTP non sono più necessari, il codice li chiude esplicitamente. Il codice chiude per ultimo il client HTTP.

## Importazioni

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

## Codice

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

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

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

// Perform work with the s3Client and dynamoDbClient.

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

## Usa **URLConnectionHttpClient** e insieme **ApacheHttpClient**

Quando si utilizza il `URLConnectionHttpClient` nella propria applicazione, è necessario fornire a ciascun client di servizio un'`URLConnectionHttpClient`istanza o un'`ApacheHttpClient`istanza utilizzando il `httpClientBuilder` metodo del service client builder.

Si verifica un'eccezione se il programma utilizza più client di servizio ed entrambe le seguenti condizioni sono vere:

- Un client di servizio è configurato per utilizzare un'`URLConnectionHttpClient`istanza
- Un altro client di servizio utilizza l'impostazione predefinita `ApacheHttpClient` senza crearla esplicitamente con i metodi `httpClient()` o `httpClientBuilder()`

L'eccezione indicherà che nel classpath sono state trovate più implementazioni HTTP.

Il seguente frammento di codice di esempio porta a un'eccezione.

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

```
        .build();

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

// Perform work with the s3Client and dynamoDbClient.

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

## Evita l'eccezione configurando esplicitamente il con un. S3Client ApacheHttpClient

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

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

// Perform work with the s3Client and dynamoDbClient.

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

### Note

Per creare esplicitamente il `ApacheHttpClient`, è necessario [aggiungere una dipendenza dall'`apache-client` artefatto](#) nel file di progetto Maven.

## Esempio di configurazione del proxy

Il seguente frammento di codice utilizza il [generatore di configurazione del proxy per il client HTTP di connessione URL](#).

```
SdkHttpClient urlHttpClient = UrlConnectionHttpClient.builder()
    .proxyConfiguration(ProxyConfiguration.builder()
        .endpoint(URI.create("http://example.com:1234"))
```



```

        .username("username")
        .password("password")
        .addNonProxyHost("localhost")
        .addNonProxyHost("host.example.com")
        .build()
    }.build();

```

Le proprietà di sistema Java equivalenti per la configurazione del proxy sono mostrate nel seguente frammento di riga di comando.

```

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

```

La configurazione equivalente che utilizza le variabili di ambiente è:

```

// Set the following environment variables.
// $ export HTTP_PROXY="http://username:password@example.com:1234"
// $ export NO_PROXY="localhost|host.example.com"

// Set the 'useSystemPropertyValues' to false on the proxy configuration.
SdkHttpClient apacheHttpClient = UrlConnectionHttpClient.builder()
    .proxyConfiguration(ProxyConfiguration.builder()
        .useSystemPropertyValues(Boolean.FALSE)
        .build())
    .build();

// Run the application.
// $ java -cp ... App

```

### Note

Il client HTTP basato su `URLConnection` attualmente non supporta le proprietà del sistema proxy HTTPS o la variabile di ambiente `HTTPS_PROXY`.

## Configurazione del client HTTP basato su Netty

Il client HTTP predefinito per le operazioni asincrone in è basato su Netty. AWS SDK for Java 2.x [NettyNioAsyncHttpClient](#) Il client basato su Netty si basa sul framework di rete asincrono basato sugli [eventi del progetto Netty](#).

[Come client HTTP alternativo, è possibile utilizzare il nuovo client HTTP basato su CRT.AWS](#) In questo argomento viene illustrato come configurare. `NettyNioAsyncHttpClient`

## Accedere a `NettyNioAsyncHttpClient`

Nella maggior parte dei casi, si utilizza la `NettyNioAsyncHttpClient` senza alcuna configurazione esplicita nei programmi asincroni. Dichiarare i tuoi client di servizio asincroni e l'SDK li configurerà automaticamente con valori standard. `NettyNioAsyncHttpClient`

Se desideri configurarli in modo esplicito `NettyNioAsyncHttpClient` o utilizzarli con più client di servizio, devi renderli disponibili per la configurazione.

Non è necessaria alcuna configurazione

Quando si dichiara una dipendenza da un client di servizio in Maven, l'SDK aggiunge una dipendenza di runtime dall'artefatto. `netty-nio-client` Ciò rende la `NettyNioAsyncHttpClient` classe disponibile per il codice in fase di esecuzione, ma non in fase di compilazione. Se non state configurando il client HTTP basato su Netty, non è necessario specificare una dipendenza per esso.

Nel seguente frammento XML di un `pom.xml` file Maven, la dipendenza dichiarata con richiamo in modo transitivo il client HTTP basato su Netty. `<artifactId>dynamodb-enhanced</artifactId>` Non è necessario dichiarare una dipendenza specifica per questo.

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

Con queste dipendenze, non è possibile apportare modifiche alla configurazione HTTP, poiché la `NettyNioAsyncHttpClient` libreria si trova solo nel classpath di runtime.

### Configurazione necessaria

Per configurare `NettyNioAsyncHttpClient`, è necessario aggiungere una dipendenza dall'`netty-nio-client` artefatto in fase di compilazione.

Fate riferimento al seguente esempio di file `pom.xml` Maven per configurare.

### NettyNioAsyncHttpClient

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

## Usa e configura il `NettyNioAsyncHttpClient`

È possibile configurare un'istanza `NettyNioAsyncHttpClient` insieme alla creazione di un client di servizio oppure configurare una singola istanza da condividere tra più client di servizio.

Con entrambi gli approcci, si utilizza [NettyNioAsyncHttpClient.Builder](#) per configurare le proprietà per l'istanza del client HTTP basata su Netty.

## Procedura consigliata: dedicare un'NettyNioAsyncHttpClientistanza a un client di servizio

Se devi configurare un'istanza di `NettyNioAsyncHttpClient`, ti consigliamo di creare un'istanza dedicata. Puoi farlo utilizzando il `httpClientBuilder` metodo del builder del client del servizio. In questo modo, il ciclo di vita del client HTTP viene gestito dall'SDK, che aiuta a evitare potenziali perdite di memoria se l'istanza non viene chiusa quando non è più necessaria.

L'esempio seguente crea un'istanza di `DynamoDbAsyncClient` che viene utilizzata da un'istanza di `DynamoDbEnhancedAsyncClient`. L'istanza di `DynamoDbAsyncClient` contiene l'istanza di `NettyNioAsyncHttpClient` con `connectionTimeout` e `maxConcurrency` i valori. L'istanza HTTP viene creata utilizzando `httpClientBuilder` il metodo di `DynamoDbAsyncClient.Builder`.

### Importazioni

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

### Codice

```
// DynamoDbAsyncClient is the lower-level client used by the enhanced client.
DynamoDbAsyncClient dynamoDbAsyncClient =
    DynamoDbAsyncClient
        .builder()
            .httpClientBuilder(NettyNioAsyncHttpClient.builder())
            .connectionTimeout(Duration.ofMillis(5_000))
            .maxConcurrency(100)
            .tlsNegotiationTimeout(Duration.ofMillis(3_500))
            .defaultsMode(DefaultsMode.IN_REGION)
            .credentialsProvider(EnvironmentVariableCredentialsProvider.create())
            .build();

// Singleton: Use dynamoDbAsyncClient and enhancedClient for all requests.
DynamoDbEnhancedAsyncClient enhancedClient =
    DynamoDbEnhancedAsyncClient
```

```

        .builder()
        .dynamoDbClient(dynamoDbAsyncClient)
        .extensions(AutoGeneratedTimestampRecordExtension.create())
        .build();

// Perform work with the dynamoDbAsyncClient and enhancedClient.

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

```

### Approccio alternativo: condividi un'**NettyNioAsyncHttpClient**istanza

Per ridurre l'utilizzo di risorse e memoria per l'applicazione, è possibile configurare un'applicazione `NettyNioAsyncHttpClient` e condividerla tra più client di servizio. Il pool di connessioni HTTP verrà condiviso, il che riduce l'utilizzo delle risorse.

#### Note

Quando un'`NettyNioAsyncHttpClient`istanza è condivisa, è necessario chiuderla quando è pronta per essere eliminata. L'SDK non chiuderà l'istanza quando il client del servizio viene chiuso.

L'esempio seguente configura un client HTTP basato su Netty utilizzato da due client di servizio. L'`NettyNioAsyncHttpClient`istanza configurata viene passata al `httpClient` metodo di ogni builder. Quando i client di servizio e il client HTTP non sono più necessari, il codice li chiude esplicitamente. Il codice chiude per ultimo il client HTTP.

### Importazioni

```

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

```

### Codice

```

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

```

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

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

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

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

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

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

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

## Esempio di configurazione del proxy

Il seguente frammento di codice utilizza il [generatore di configurazione proxy per il client HTTP Netty](#).

```
SdkAsyncHttpClient nettyHttpClient = NettyNioAsyncHttpClient.builder()
    .proxyConfiguration(ProxyConfiguration.builder()
        .scheme("https")
        .host("myproxy")
        .port(1234)
        .username("username")
        .password("password")
        .nonProxyHosts(Set.of("localhost", "host.example.com")))
        .build())
    .build();
```

Le proprietà di sistema Java equivalenti per la configurazione del proxy sono mostrate nel seguente frammento di riga di comando.

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

### Important

Per utilizzare una qualsiasi delle proprietà del sistema proxy HTTPS, la scheme proprietà deve essere impostata nel codice su. `https` Se la proprietà `scheme` non è impostata nel codice, per impostazione predefinita lo schema è `HTTP` e l'SDK cerca `http.*` solo le proprietà di sistema.

La configurazione equivalente che utilizza le variabili di ambiente è:

```
// Set the following environment variables.
// $ export HTTPS_PROXY="https://username:password@myproxy:1234"
// $ export NO_PROXY="localhost|host.example.com"

// Set the 'useSystemPropertyValues' to false on the proxy configuration.
SdkAsyncHttpClient nettyHttpClient = NettyNioAsyncHttpClient.builder()
    .proxyConfiguration(ProxyConfiguration.builder()
        .useSystemPropertyValues(Boolean.FALSE)
        .build())
    .build();

// Run the application.
// $ java -cp ... App
```

## Configurazione AWS dei client HTTP basati su CRT

I client HTTP AWS basati su CRT includono quelli sincroni e asincroni.

[AwsCrtHttpClient](#) e [AwsCrtAsyncHttpClient](#) I client HTTP AWS basati su CRT offrono i seguenti vantaggi del client HTTP:

- Tempo di avvio SDK più rapido
- Ingombro di memoria ridotto

- Tempo di latenza ridotto
- Gestione dello stato della connessione
- Bilanciamento del carico DNS

## AWS Componenti basati su CRT nell'SDK

I client HTTP AWS basati su CRT, descritti in questo argomento, e il client S3 AWS basato su CRT sono componenti diversi dell'SDK.

I client HTTP sincroni e asincroni AWS basati su CRT sono interfacce client HTTP SDK di implementazione e vengono utilizzati per comunicazioni HTTP generali. Sono alternative agli altri client HTTP sincroni o asincroni dell'SDK con vantaggi aggiuntivi.

Il [client S3 AWS basato su CRT](#) è un'implementazione dell'AsyncClientinterfaccia [S3](#) e viene utilizzato per lavorare con il servizio Amazon S3. È un'alternativa all'implementazione dell'interfaccia basata su Java e offre diversi vantaggi. `S3AsyncClient`

Sebbene entrambi i componenti utilizzino librerie del [AWS Common Runtime](#), i client HTTP AWS basati su CRT non utilizzano la [libreria aws-c-s 3](#) e non supportano le funzionalità dell'API di caricamento multipart di [S3](#). Il client S3 AWS basato su CRT, invece, è stato creato appositamente per supportare le funzionalità dell'API di caricamento multipart di S3.

## Accedi AWS ai client HTTP basati su CRT

Prima di poter utilizzare i client HTTP AWS basati su CRT, aggiungete l'`aws-crt-client` artefatto con una versione minima di 2.22.0 alle dipendenze del progetto.

Il seguente Maven `pom.xml` mostra i client HTTP AWS basati su CRT dichiarati utilizzando il meccanismo della distinta base (BOM).

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



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

[Visita il repository centrale di Maven per la versione più recente.](#)

## Usa e configura un AWS client HTTP basato su CRT

È possibile configurare un client HTTP AWS basato su CRT insieme alla creazione di un client di servizio oppure è possibile configurare una singola istanza da condividere tra più client di servizio.

Con entrambi gli approcci, si utilizza un builder per [configurare le proprietà per l'istanza del client HTTP basato su AWS CRT](#).

Procedura consigliata: dedicare un'istanza a un client di servizio

Se devi configurare un'istanza di un client HTTP AWS basato su CRT, ti consigliamo di dedicare l'istanza costruendola insieme al client di servizio. Puoi farlo utilizzando il `httpClientBuilder` metodo del builder del client del servizio. In questo modo, il ciclo di vita del client HTTP viene gestito dall'SDK, che aiuta a evitare potenziali perdite di memoria se l'istanza del client HTTP AWS basato su CRT non viene chiusa quando non è più necessaria.

L'esempio seguente crea un client di servizio S3 e configura un client HTTP basato su CRT con valori e. `AWS connectionTimeout` `maxConcurrency`

### Synchronous client

#### Importazioni

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

#### Codice

```
// Singleton: Use s3Client for all requests.
S3Client s3Client = S3Client.builder()
    .httpClientBuilder(AwsCrtHttpClient
        .builder()
        .connectionTimeout(Duration.ofSeconds(3))
        .maxConcurrency(100))
    .build();

// Perform work with the s3Client.

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

## Asynchronous client

### Importazioni

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

### Codice

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

// Perform work with the s3AsyncClient.

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

## Approccio alternativo: condividi un'istanza

Per ridurre l'utilizzo di risorse e memoria per l'applicazione, è possibile configurare un client HTTP AWS basato su CRT e condividerlo tra più client di servizio. Il pool di connessioni HTTP verrà condiviso, il che riduce l'utilizzo delle risorse.

**Note**

Quando un'istanza client HTTP AWS basata su CRT viene condivisa, è necessario chiuderla quando è pronta per essere eliminata. L'SDK non chiuderà l'istanza quando il client del servizio viene chiuso.

L'esempio seguente configura un'istanza client HTTP AWS basata su CRT con valori e. `connectionTimeout` `maxConcurrency` L'istanza configurata viene passata al `httpClient` metodo del builder di ogni client di servizio. Quando i client di servizio e il client HTTP non sono più necessari, vengono chiusi in modo esplicito. Il client HTTP viene chiuso per ultimo.

**Synchronous client****Importazioni**

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

**Codice**

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

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

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

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

## Asynchronous client

### Importazioni

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

### Codice

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

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

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

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

## Imposta un client HTTP AWS basato su CRT come predefinito

Puoi configurare il tuo file di build Maven in modo che l'SDK utilizzi un client HTTP AWS basato su CRT come client HTTP predefinito per i client di servizio.

A tale scopo, è possibile aggiungere un `exclusions` elemento con le dipendenze del client HTTP predefinite a ciascun elemento del client di servizio.

Nell'`pom.xml` esempio seguente, l'SDK utilizza un client HTTP AWS basato su CRT per i servizi S3. Se il client di servizio nel codice è un `S3AsyncClient`, l'SDK lo utilizza. `AwsCrtAsyncHttpClient`. Se il client di servizio è un `S3Client`, l'SDK utilizza. `AwsCrtHttpClient`. Con questa configurazione il client HTTP asincrono predefinito basato su Netty e l'HTTP sincrono predefinito basato su Apache non sono disponibili.

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

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

[Visita l'archivio centrale di Maven per il valore VERSION più recente.](#)

#### Note

Se in un pom.xml file vengono dichiarati più client di servizio, tutti richiedono l'exclusionselemento XML.

Utilizzate una proprietà di sistema Java

Per utilizzare i client HTTP AWS basati su CRT come HTTP predefinito per l'applicazione, è possibile impostare la proprietà del sistema Java su un valore `software.amazon.awssdk.http.async.service.impl` di `software.amazon.awssdk.http.crt.AwsCrtSdkHttpClient`

Per impostarlo durante l'avvio dell'applicazione, eseguite un comando simile al seguente.

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

Utilizzate il seguente frammento di codice per impostare la proprietà di sistema nel codice dell'applicazione.

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

**Note**

È necessario aggiungere una dipendenza dall'elemento nel `pom1.xml` file quando si utilizza una proprietà di sistema per configurare l'uso dei client HTTP basati su `aws-crt-client` CRT. AWS

## Configurazione avanzata dei client HTTP basati su CRT AWS

È possibile utilizzare varie impostazioni di configurazione dei client HTTP AWS basati su CRT, tra cui la configurazione dello stato della connessione e il tempo massimo di inattività. È possibile esaminare le [opzioni di configurazione disponibili per](#) `AwsCrtAsyncHttpClient` È possibile configurare le stesse opzioni per `AwsCrtHttpClient`.

### Configurazione dello stato della connessione

È possibile configurare la configurazione dello stato della connessione per i client HTTP AWS basati su CRT utilizzando il `connectionHealthConfiguration` metodo del generatore di client HTTP.

L'esempio seguente crea un client di servizio S3 che utilizza un'istanza del client HTTP AWS basata su CRT configurata con la configurazione dello stato della connessione e un tempo di inattività massimo per le connessioni.

### Synchronous client

#### Importazioni

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

#### Codice

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

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

// Perform work with s3Client.

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

## Asynchronous client

### Importazioni

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

### Codice

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

// Perform work with s3AsyncClient.

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

## Supporto HTTP/2

Il protocollo HTTP/2 non è ancora supportato nei client HTTP AWS basati su CRT, ma è previsto per una versione futura.

Nel frattempo, se utilizzi client di servizio che richiedono il supporto HTTP/2 come `the o the`, considera di utilizzare invece il [KinesisAsyncClient](#), [TranscribeStreamingAsyncClient](#) o [NettyNioAsyncHttpClient](#).



## Esempio di configurazione del proxy

Il seguente frammento di codice mostra l'uso di [ProxyConfiguration.Builder](#) ciò che si utilizza per configurare l'impostazione del proxy nel codice.

### Synchronous client

#### Importazioni

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

#### Codice

```
SdkHttpClient crtHttpClient = AwsCrtHttpClient.builder()
    .proxyConfiguration(ProxyConfiguration.builder()
        .scheme("https")
        .host("myproxy")
        .port(1234)
        .username("username")
        .password("password")
        .nonProxyHosts(Set.of("localhost", "host.example.com")))
        .build())
    .build();
```

### Asynchronous client

#### Importazioni

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

#### Codice

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

```
.port(1234)
.username("username")
.password("password")
.nonProxyHosts(Set.of("localhost", "host.example.com"))
.build()
.build();
```

Le proprietà di sistema Java equivalenti per la configurazione del proxy sono mostrate nel seguente frammento di riga di comando.

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

### Important

Per utilizzare una qualsiasi delle proprietà del sistema proxy HTTPS, la `scheme` proprietà deve essere impostata nel codice su `https`. Se la proprietà `scheme` non è impostata nel codice, per impostazione predefinita lo schema è `HTTP` e l'SDK cerca `http.*` solo le proprietà di sistema.

La configurazione equivalente che utilizza le variabili di ambiente è:

```
// Set the following environment variables.
// $ export HTTPS_PROXY="https://username:password@myproxy:1234"
// $ export NO_PROXY="localhost|host.example.com"

// Set the 'useSystemPropertyValues' to false on the proxy configuration.
SdkAsyncHttpClient crtAsyncHttpClient = AwsCrtAsyncHttpClient.builder()
    .proxyConfiguration(ProxyConfiguration.builder()
        .scheme("https")
        .useSystemPropertyValues(Boolean.FALSE)
        .build())
    .build();

// Run the application.
// $ java -cp ... App
```

## Configurazione dei proxy HTTP

È possibile configurare i proxy HTTP utilizzando il codice, impostando le proprietà del sistema Java o impostando le variabili di ambiente.

### Configura nel codice

I proxy vengono configurati in codice con un `ProxyConfiguration` builder specifico per il client quando si crea il client di servizio. Il codice seguente mostra un esempio di configurazione proxy per un client HTTP basato su Apache utilizzato da un client di servizio Amazon S3.

```
SdkHttpClient httpClient1 = ApacheHttpClient.builder()
    .proxyConfiguration(ProxyConfiguration.builder()
        .endpoint(URI.create("http://proxy.example.com"))
        .username("username")
        .password("password")
        .addNonProxyHost("localhost")
        .build())
    .build();

S3Client s3Client = S3Client.builder()
    .httpClient(httpClient)
    .build();
```

La sezione relativa a ciascun client HTTP di questo argomento mostra un esempio di configurazione proxy.

- [Client Apache HTTP](#)
- [Client HTTP basato su URLConnection](#)
- [Client HTTP basato su Netty](#)
- [AWS Client HTTP basato su CRT](#)

### Configura i proxy HTTP con impostazioni esterne

Anche se non utilizzi esplicitamente un `ProxyConfiguration` builder nel codice, l'SDK cerca le impostazioni esterne per configurare una configurazione proxy predefinita.

Per impostazione predefinita, l'SDK cerca innanzitutto le proprietà del sistema JVM. Se viene trovata anche una sola proprietà, l'SDK utilizza il valore e qualsiasi altro valore di proprietà di sistema. Se non sono disponibili proprietà di sistema, l'SDK cerca le variabili di ambiente proxy.

L'SDK può utilizzare le seguenti proprietà di sistema e variabili di ambiente Java.

### Proprietà del sistema Java

Proprietà del sistema	Descrizione	Supporto client HTTP
http.ProxyHost	Nome host del server proxy HTTP	Tutti
HTTP.ProxyPort	Numero di porta del server proxy HTTP	Tutti
http.proxyUser	Nome utente per l'autenticazione del proxy HTTP	Tutti
Password http.proxy	Password per l'autenticazione del proxy HTTP	Tutti
http.nonProxyHosts	Elenco di host che devono essere raggiunti direttamente, ignorando il proxy. <a href="#">Questo elenco è valido anche quando si utilizza HTTPS.</a>	Tutti
https://proxyHost	Nome host del server proxy HTTPS	Netty, CRT
Porta https://Proxy_Port	Numero di porta del server proxy HTTPS	Netty, CRT
Utente https://proxy	Nome utente per l'autenticazione del proxy HTTPS	Netty, CRT
Password https://proxy/	Password per l'autenticazione del proxy HTTPS	Netty, CRT

## Variabili di ambiente

Variabile di ambiente	Descrizione	Supporto client HTTP
HTTP_PROXY <sup>1</sup>	Un URL valido con uno schema HTTP	Tutti
HTTPS_PROXY <sup>1</sup>	Un URL valido con uno schema HTTPS	Netty, CRT
NO_PROXY <sup>2</sup>	Elenco di host che devono essere raggiunti direttamente, ignorando il proxy. L'elenco è valido sia per HTTP che per HTTPS.	Tutti

Visualizza la chiave e le note a piè di pagina

Tutti: tutti i client HTTP offerti dall'SDK

—`URLConnectionHttpClient`, `ApacheHttpClient`, `NettyNioAsyncHttpClient`.  
`AwsCrtAsyncHttpClient`

Netty: il client HTTP basato su Netty (`NettyNioAsyncHttpClient`)

CRT - I client HTTP AWS basati su CRT, (`AwsCrtHttpClient` `AwsCrtAsyncHttpClient`)

<sup>1</sup> La variabile di ambiente richiesta, indipendentemente dal fatto che sia `HTTP_PROXY` o `HTTPS_PROXY`, dipende dall'impostazione dello schema nel client. `ProxyConfiguration` Lo schema predefinito è HTTP. Il seguente frammento mostra come modificare lo schema in HTTPS utilizzato per la risoluzione delle variabili di ambiente.

```
SdkHttpClient httpClient = ApacheHttpClient.builder()
    .proxyConfiguration(ProxyConfiguration.builder()
        .scheme("https")
        .build())
    .build();
```

<sup>2</sup> La variabile di `NO_PROXY` ambiente supporta una combinazione di separatori `|` e `,` tra i nomi host. I nomi host possono includere il carattere jolly `*`.

## Usa una combinazione di impostazioni

È possibile utilizzare una combinazione di impostazioni del proxy HTTP nel codice, nelle proprietà di sistema e nelle variabili di ambiente.

Example — configurazione fornita da una proprietà di sistema e dal codice

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

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

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

L'SDK risolve le seguenti impostazioni proxy.

```
Host = localhost
Port = 1234
Password = SYS_PROP_password
UserName = username
Non ProxyHost = null
```

Example — sono disponibili sia le proprietà di sistema che le variabili di ambiente

Il `ProxyConfiguration` builder di ogni client HTTP offre impostazioni denominate `useSystemPropertyValues` e `useEnvironmentVariablesValues`. Per impostazione predefinita, entrambe le impostazioni sono `true`. Quando `true`, l'SDK utilizza automaticamente i valori delle proprietà di sistema o delle variabili di ambiente per opzioni non fornite dal `ProxyConfiguration` builder.

**⚠ Important**

Le proprietà di sistema hanno la precedenza sulle variabili di ambiente. Se viene trovata una proprietà del sistema proxy HTTP, l'SDK recupera tutti i valori dalle proprietà di sistema e nessuno dalle variabili di ambiente. Se desideri dare priorità alle variabili di ambiente rispetto alle proprietà di sistema, imposta su. `useSystemPropertyValues false`

Per questo esempio, le seguenti impostazioni sono disponibili in fase di esecuzione:

```
// System properties
http.proxyHost=SYS_PROP_HOST.com
http.proxyPort=2222
http.password=SYS_PROP_PASSWORD
http.user=SYS_PROP_USER

// Environment variables
HTTP_PROXY="http://EnvironmentUser:EnvironmentPassword@ENV_VAR_HOST:3333"
NO_PROXY="environmentnonproxy.host,environmentnonproxy2.host:1234"
```

Il client di servizio viene creato con una delle seguenti istruzioni. Nessuna delle istruzioni imposta esplicitamente un'impostazione proxy.

```
DynamoDbClient client = DynamoDbClient.create();
DynamoDbClient client = DynamoDbClient.builder().build();
DynamoDbClient client = DynamoDbClient.builder()
    .httpClient(ApacheHttpClient.builder()
        .proxyConfiguration(ProxyConfiguration.builder()
            .build())
        .build())
    .build();
```

Le seguenti impostazioni proxy vengono risolte dall'SDK:

```
Host = SYS_PROP_HOST.com
Port = 2222
Password = SYS_PROP_PASSWORD
UserName = SYS_PROP_USER
Non ProxyHost = null
```

Poiché il client del servizio dispone di impostazioni proxy predefinite, l'SDK cerca le proprietà del sistema e quindi le variabili di ambiente. Poiché le impostazioni delle proprietà di sistema hanno la precedenza sulle variabili di ambiente, l'SDK utilizza solo le proprietà di sistema.

Se l'uso delle proprietà di sistema viene modificato `false` come mostrato nel codice seguente, l'SDK risolve solo le variabili di ambiente.

```
DynamoDbClient client = DynamoDbClient.builder()
    .httpClient(ApacheHttpClient.builder()
        .proxyConfiguration(ProxyConfiguration.builder()
            .useSystemPropertyValues(Boolean.FALSE)
            .build())
        .build())
    .build();
```

Le impostazioni proxy risolte tramite HTTP sono:

```
Host = ENV_VAR_HOST
Port = 3333
Password = EnvironmentPassword
UserName = EnvironmentUser
Non ProxyHost = environmentnonproxy.host, environmentnonproxy2.host:1234
```

## Gestione delle eccezioni per AWS SDK for Java 2.x

Comprendere come e quando AWS SDK for Java 2.x genera eccezioni è importante per la creazione di applicazioni di alta qualità utilizzando l'SDK. Nelle seguenti sezioni vengono descritti i diversi casi di eccezioni che vengono generate dall'SDK e come gestirle in modo appropriato.

### Perché le eccezioni non controllate?

AWS SDK for Java utilizza eccezioni runtime (o non controllate) anziché eccezioni controllate per i seguenti motivi:

- Per consentire agli sviluppatori di controllare ogni dettaglio degli errori che desiderano gestire senza costringerli a gestire casi eccezionali che non destino preoccupazione (rendendo il codice eccessivamente dettagliato)
- Per prevenire problemi di scalabilità intrinseca con eccezioni controllate in applicazioni di grandi dimensioni



In generale, le eccezioni controllate funzionano bene su scale ridotte, ma possono diventare problematiche all'aumentare delle dimensioni e della complessità delle applicazioni.

## AwsServiceException (e sottoclassi)

[AwsServiceException](#) è l'eccezione più comune che si verificherà quando si utilizza. AWS SDK for Java [AwsServiceException](#) è una sottoclasse di quella più generale [SdkServiceException](#). [AwsServiceExceptions](#) rappresenta una risposta di errore da un servizio AWS. Ad esempio, se provi a terminare un'Amazon EC2 istanza che non esiste, Amazon EC2 restituirà una risposta di errore e tutti i dettagli di tale risposta di errore verranno inclusi nella [AwsServiceException](#) risposta generata.

Quando incontri un [AwsServiceException](#), sai che la tua richiesta è stata inviata con successo a Servizio AWS ma non può essere elaborata correttamente. Questo può essere dovuto a errori nei parametri della richiesta o a errori sul lato servizio.

[AwsServiceException](#) fornisce informazioni quali:

- Codice di stato HTTP restituito
- Codice AWS di errore restituito
- Messaggio di errore dettagliato del servizio della [AwsErrorDetails](#) classe
- AWSID della richiesta non riuscita

Nella maggior parte dei casi, [AwsServiceException](#) viene generata una sottoclasse specifica del servizio di per consentire agli sviluppatori un controllo granulare sulla gestione dei casi di errore tramite i block catch. Il riferimento all'API Java SDK per visualizzare il gran numero di sottoclassi. [AwsServiceException](#) Utilizza i collegamenti alle sottoclassi per visualizzare in dettaglio le eccezioni granulari generate da un servizio.

Ad esempio, i seguenti collegamenti al riferimento dell'API SDK mostrano le gerarchie delle eccezioni per alcune comuni. Servizi AWS L'elenco delle sottoclassi mostrato in ogni pagina mostra le eccezioni specifiche che il codice può catturare.

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

Per ulteriori informazioni su un'eccezione, ispeziona l'`errorCode` oggetto. [AwsErrorDetails](#) È possibile utilizzare il `errorCode` valore per cercare informazioni nell'API della guida ai servizi. Ad esempio, se `S3Exception` viene rilevato un e il `AwsErrorDetails#errorCode()` valore è `InvalidRequest`, utilizza l'[elenco dei codici di errore](#) nell'Amazon S3 API Reference per visualizzare ulteriori dettagli.

## SdkClientException

[SdkClientException](#) indica che si è verificato un problema all'interno del codice client Java, durante il tentativo di inviare una richiesta a AWS o durante il tentativo di analizzare una risposta da AWS. Un `SdkClientException` è generalmente più grave di un `SdkServiceException` e indica un problema importante che impedisce al client di effettuare chiamate di servizio ai AWS servizi. Ad esempio, AWS SDK for Java genera una `SdkClientException` se una connessione di rete non è disponibile quando si cerca di chiamare un'operazione su uno dei client.

## Eccezioni e comportamenti relativi ai tentativi

L'SDK for Java ritenta le richieste per [diverse eccezioni lato client](#) e [per i codici di stato HTTP](#) che riceve dalle risposte. Servizio AWS Questi errori vengono gestiti come parte dell'eredità utilizzata dai client di servizio per impostazione predefinita `RetryMode`. Il riferimento all'API Java per [RetryMode](#) descrive i vari modi in cui è possibile configurare la modalità.

Per personalizzare le eccezioni e i codici di stato HTTP che attivano i nuovi tentativi automatici, configura il client di servizio con un programma [RetryPolicy](#) che aggiunge [RetryOnExceptionsCondition](#) istanze [RetryOnStatusCodeCondition](#).

## Registrazione con l'SDK for Java 2.x

AWS SDK for Java 2.x Utilizza [SLF4J](#), che è un livello di astrazione che consente l'uso di uno qualsiasi dei diversi sistemi di registrazione in fase di esecuzione.

[I sistemi di registrazione supportati includono Java Logging Framework e Apache Log4j 2, tra gli altri.](#) Questo argomento mostra come utilizzare Log4j 2 come sistema di registrazione per lavorare con l'SDK.

## File di configurazione Log4j 2

In genere si utilizza un file di configurazione, denominato `log4j2.xml`. File di configurazione di esempio sono mostrati di seguito. Per ulteriori informazioni sui valori utilizzati nel file di configurazione, consulta il [manuale per la configurazione di Log4j](#).

Il `log4j2.xml` file deve trovarsi nel classpath all'avvio dell'applicazione. Per un progetto Maven, inserisci il file nella directory. `<project-dir>/src/main/resources`

[Il file di `log4j2.xml` configurazione specifica proprietà come il livello di registrazione, a cui viene inviato l'output di registrazione \(ad esempio, a un file o alla console\) e il formato dell'output.](#) Il livello di registrazione specifica il livello di dettaglio generato da Log4j 2. [Log4j 2 supporta il concetto di gerarchie di registrazione multiple.](#) Il livello di registrazione è impostato in modo indipendente per ogni gerarchia. La gerarchia di registrazione principale utilizzata con è. `software.amazon.awssdk`

## Aggiungi dipendenza dalla registrazione

Per configurare l'associazione Log4j 2 per SLF4J nel tuo file di build, usa quanto segue.

### Maven

Aggiungi i seguenti elementi al tuo file. `pom.xml`

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

### Gradle–Kotlin DSL

Aggiungi quanto segue al tuo `build.gradle.kts` file.

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

```
    ...  
}  
    ...
```

Usa `2.20.0` per la versione minima dell'`log4j-slf4j2-impl` artefatto. Per la versione più recente, usa la versione pubblicata su [Maven Central](#). Sostituisci *VERSION* con la versione che utilizzerai.

## Errori e avvisi specifici dell'SDK

Ti consigliamo di lasciare sempre la gerarchia dei logger «`software.amazon.awssdk`» impostata su «`WARN`» per catturare eventuali messaggi importanti dalle librerie client dell'SDK. Ad esempio, se il client Amazon S3 rileva che l'applicazione non ha chiuso correttamente `InputStream` e che potrebbe esserci una perdita di risorse, il client S3 lo segnala tramite un messaggio di avviso ai log. Questo garantisce inoltre che i messaggi vengono registrati se il client presenta problemi di gestione delle richieste o delle risposte.

Il `log4j2.xml` file seguente lo imposta su «`WARN`», che genera l'output di messaggi di avviso e `rootLogger` a livello di errore da tutti i logger dell'applicazione, inclusi quelli nella gerarchia «`software.amazon.awssdk`». In alternativa, puoi impostare esplicitamente la gerarchia dei logger «`software.amazon.awssdk`» su «`WARN`», se utilizzata. `<Root level="ERROR">`

### Esempio di file di configurazione Log4j2.xml

Questa configurazione registrerà i messaggi ai livelli «`ERROR`» e «`WARN`» nella console per tutte le gerarchie di logger.

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

## Registrazione riassuntiva di richieste/risposte

Ogni richiesta a un Servizio AWS genera un ID di AWS richiesta univoco, utile in caso di problemi relativi alla gestione di Servizio AWS una richiesta. AWS Gli ID di richiesta sono accessibili a livello di codice tramite [SdkServiceException](#) gli oggetti dell'SDK per qualsiasi chiamata di servizio fallita e possono anche essere segnalati tramite il livello di registro «DEBUG» del logger «software.amazon.awssdk.request».

Il file seguente consente un riepilogo delle richieste e delle risposte. `log4j2.xml`

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

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

Di seguito è riportato un esempio di output del log:

```
2022-09-23 16:02:08 [main] DEBUG software.amazon.awssdk.request:85 - Sending Request:
DefaultSdkHttpFullRequest(httpMethod=POST, protocol=https, host=dynamodb.us-
east-1.amazonaws.com, encodedPath=/, headers=[amz-sdk-invocation-id, Content-Length,
Content-Type, User-Agent, X-Amz-Target], queryParameters=[])
2022-09-23 16:02:08 [main] DEBUG software.amazon.awssdk.request:85 - Received
successful response: 200, Request ID:
QS9DUMME2NHEDH8TGT9N5V530JVV4KQNS05AEMVJF66Q9ASUAAJG, Extended Request ID: not
available
```

Se sei interessato solo all'ID della richiesta, usa `<Logger name="software.amazon.awssdk.requestId" level="DEBUG" />`.

## Registrazione SDK a livello di debug

Se hai bisogno di maggiori dettagli su ciò che sta facendo l'SDK, puoi impostare il livello di registrazione del logger su `software.amazon.awssdk.DEBUG`. A questo livello, l'SDK genera una grande quantità di dettagli, quindi ti consigliamo di impostare questo livello per risolvere gli errori utilizzando i test di integrazione.

A questo livello di registrazione, l'SDK registra informazioni sulla configurazione, la risoluzione delle credenziali, gli intercettori di esecuzione, l'attività TLS di alto livello, la firma delle richieste e molto altro.

Di seguito è riportato un esempio di istruzioni emesse dall'SDK a livello di chiamata `DEBUG S3Client#listBuckets()`

```
DEBUG s.a.a.r.p.AwsRegionProviderChain:57 - Unable to load region from
software.amazon.awssdk.regions.providers.SystemSettingsRegionProvider@324dcd31:Unable
to load region from system settings. Region must be specified either via environment
variable (AWS_REGION) or system property (aws.region).
DEBUG s.a.a.c.i.h.l.ClasspathSdkHttpServiceProvider:85 - The HTTP implementation loaded
is software.amazon.awssdk.http.apache.ApacheSdkHttpService@a23a01d
DEBUG s.a.a.c.i.ExecutionInterceptorChain:85 - Creating an interceptor
chain that will apply interceptors in the following order:
[software.amazon.awssdk.core.internal.interceptor.HttpChecksumValidationInterceptor@69b2f8e5,
software.amazon.awssdk.awscore.interceptor.HelpfulUnknownHostExceptionInterceptor@6331250e,
software.amazon.awssdk.awscore.eventstream.EventStreamInitialRequestInterceptor@a10c1b5,
software.amazon.awssdk.awscore.interceptor.TraceIdExecutionInterceptor@644abb8f,
software.amazon.awssdk.services.s3.auth.scheme.internal.S3AuthSchemeInterceptor@1a411233,
software.amazon.awssdk.services.s3.endpoints.internal.S3ResolveEndpointInterceptor@70325d20,
software.amazon.awssdk.services.s3.endpoints.internal.S3RequestSetEndpointInterceptor@7c2327fa,
software.amazon.awssdk.services.s3.internal.handlers.StreamingRequestInterceptor@4d847d32,
software.amazon.awssdk.services.s3.internal.handlers.CreateBucketInterceptor@5f462e3b,
software.amazon.awssdk.services.s3.internal.handlers.CreateMultipartUploadRequestInterceptor@3
software.amazon.awssdk.services.s3.internal.handlers.DecodeUrlEncodedResponseInterceptor@58065
software.amazon.awssdk.services.s3.internal.handlers.GetBucketPolicyInterceptor@3605c4d3,
software.amazon.awssdk.services.s3.internal.handlers.S3ExpressChecksumInterceptor@585c13de,
software.amazon.awssdk.services.s3.internal.handlers.AsyncChecksumValidationInterceptor@187eb9
software.amazon.awssdk.services.s3.internal.handlers.SyncChecksumValidationInterceptor@726a6b9
software.amazon.awssdk.services.s3.internal.handlers.EnableTrailingChecksumInterceptor@6ad11a5
software.amazon.awssdk.services.s3.internal.handlers.ExceptionTranslationInterceptor@522b2631,
software.amazon.awssdk.services.s3.internal.handlers.GetObjectInterceptor@3ff57625,
software.amazon.awssdk.services.s3.internal.handlers.CopySourceInterceptor@1ee29c84,
software.amazon.awssdk.services.s3.internal.handlers.ObjectMetadataInterceptor@7c8326a4]
```

```
DEBUG s.a.a.u.c.CachedSupplier:85 - (SsoOidcTokenProvider()) Cached value is stale and
will be refreshed.
...
DEBUG s.a.a.c.i.ExecutionInterceptorChain:85 - Creating an interceptor
chain that will apply interceptors in the following order:
[software.amazon.awssdk.core.internal.interceptor.HttpChecksumValidationInterceptor@51351f28,
software.amazon.awssdk.awscore.interceptor.HelpfulUnknownHostExceptionInterceptor@21618fa7,
software.amazon.awssdk.awscore.eventstream.EventStreamInitialRequestInterceptor@15f2eda3,
software.amazon.awssdk.awscore.interceptor.TraceIdExecutionInterceptor@34cf294c,
software.amazon.awssdk.services.sso.auth.scheme.internal.SsoAuthSchemeInterceptor@4d7aaca2,
software.amazon.awssdk.services.sso.endpoints.internal.SsoResolveEndpointInterceptor@604b1e1d,
software.amazon.awssdk.services.sso.endpoints.internal.SsoRequestSetEndpointInterceptor@625668
...
DEBUG s.a.a.request:85 - Sending Request: DefaultSdkHttpRequest(httpMethod=GET,
protocol=https, host=portal.sso.us-east-1.amazonaws.com, encodedPath=/federation/
credentials, headers=[amz-sdk-invocation-id, User-Agent, x-amz-sso_bearer_token],
queryParameters=[role_name, account_id])
DEBUG s.a.a.c.i.h.p.s.SigningStage:85 - Using SelectedAuthScheme: smithy.api#noAuth
DEBUG s.a.a.h.a.i.c.SdkTlsSocketFactory:366 - Connecting socket to portal.sso.us-
east-1.amazonaws.com/18.235.195.183:443 with timeout 2000
...
DEBUG s.a.a.requestId:85 - Received successful response: 200, Request ID: bb4f40f4-
e920-4b5c-8648-58f26e7e08cd, Extended Request ID: not available
DEBUG s.a.a.request:85 - Received successful response: 200, Request ID: bb4f40f4-
e920-4b5c-8648-58f26e7e08cd, Extended Request ID: not available
DEBUG s.a.a.u.c.CachedSupplier:85 -
  (software.amazon.awssdk.services.sso.auth.SsoCredentialsProvider@b965857) Successfully
  refreshed cached value. Next Prefetch Time: 2024-04-25T22:03:10.097Z. Next Stale Time:
  2024-04-25T22:05:30Z
DEBUG s.a.a.c.i.ExecutionInterceptorChain:85 - Interceptor
'software.amazon.awssdk.services.s3.endpoints.internal.S3RequestSetEndpointInterceptor@7c2327f
modified the message with its modifyHttpRequest method.
...
DEBUG s.a.a.c.i.h.p.s.SigningStage:85 - Using SelectedAuthScheme: aws.auth#sigv4
...
DEBUG s.a.a.a.s.Aws4Signer:85 - AWS4 Canonical Request: GET
...
DEBUG s.a.a.h.a.a.i.s.DefaultV4RequestSigner:85 - AWS4 String to sign: AWS4-HMAC-SHA256
20240425T210631Z
20240425/us-east-1/s3/aws4_request
aafb7784627fa7a49584256cb746279751c48c2076f813259ef767ecce304d64
DEBUG s.a.a.h.a.i.c.SdkTlsSocketFactory:366 - Connecting socket to s3.us-
east-1.amazonaws.com/52.217.41.86:443 with timeout 2000
```

...

Il `log4j2.xml` file seguente configura l'output precedente.

```
<Configuration status="WARN">
  <Appenders>
    <Console name="ConsoleAppender" target="SYSTEM_OUT">
      <PatternLayout pattern="%-5p %c{1.}:%L - %m%n" />
    </Console>
  </Appenders>

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

## Abilita la registrazione dei cavi

Può essere utile visualizzare le richieste e le risposte esatte che l'SDK for Java 2.x invia e riceve. Se è necessario accedere a queste informazioni, è possibile abilitarle temporaneamente aggiungendo la configurazione necessaria in base al client HTTP utilizzato dal client di servizio.

Per impostazione predefinita, i client di servizio sincroni, come [S3Client](#), utilizzano un [Apache sottostante, mentre i client](#) di servizio asincroni `HttpClient`, come [S3 AsyncClient](#), utilizzano un client HTTP non bloccante `Netty`.

Di seguito è riportato un elenco dettagliato dei client HTTP che è possibile utilizzare per le due categorie di client di servizio:

Client HTTP sincroni	Client HTTP asincroni
<a href="#">ApacheHttpClient</a> (impostazione predefinita)	<a href="#">NettyNioAsyncHttpClient</a> (impostazione predefinita)
<a href="#">URLConnectionHttpClient</a>	<a href="#">AwsCrtAsyncHttpClient</a>



Consultate la scheda appropriata riportata di seguito per le impostazioni di configurazione da aggiungere in base al client HTTP sottostante.

### Warning

Ti consigliamo di usare la registrazione in rete solo per scopi di debug. È opportuno disabilitarla negli ambienti di produzione in quanto può registrare dati sensibili. L'intera richiesta o risposta senza crittografia viene registrata, anche per una chiamata HTTPS. Per richieste di grandi dimensioni (ad esempio, per caricare un file Amazon S3) o risposte, anche il verbose wire logging può influire in modo significativo sulle prestazioni dell'applicazione.

## ApacheHttpClient

Aggiungi il logger «org.apache.http.wire» al file di `log4j2.xml` configurazione e imposta il livello su «DEBUG».

Il seguente `log4j2.xml` file attiva la registrazione cablata completa per Apache. HttpClient

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

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

È necessaria un'ulteriore dipendenza di Maven dall'`log4j-1.2-api` artefatto per la registrazione via cavo con Apache, poiché utilizza 1.2 sotto il cofano.

Il set completo di dipendenze Maven per `log4j 2`, incluso il wire logging per il client HTTP Apache, è mostrato nei seguenti frammenti di file di build.

## Maven

```

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

<!-- The following is needed for Apache HttpClient wire logging -->
<dependency>
  <groupId>org.apache.logging.log4j</groupId>
  <artifactId>log4j-1.2-api</artifactId>
</dependency>
...

```

## DSL Gradle-Kotlin

```

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

```

Usa `2.20.0` per la versione minima dell'artefatto. `log4j-bom` Per la versione più recente, usa la versione pubblicata su [Maven Central](#). Sostituisci `VERSION` con la versione che utilizzerai.

## URLConnectionHttpClient

Per registrare i dettagli dei client di servizio che utilizzano il `URLConnectionHttpClient`, crea innanzitutto un `logging.properties` file con i seguenti contenuti:

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

Imposta la seguente proprietà del sistema JVM con il percorso completo di:

`logging.properties`

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

Questa configurazione registrerà solo le intestazioni della richiesta e della risposta, ad esempio:

```
<Request> FINE: sun.net.www.MessageHeader@35a9782c11 pairs: {GET /fileuploadtest
HTTP/1.1: null}{amz-sdk-invocation-id: 5f7e707e-4ac5-bef5-ba62-00d71034ffdc}
{amz-sdk-request: attempt=1; max=4}{Authorization: AWS4-HMAC-SHA256
Credential=<deleted>/20220927/us-east-1/s3/aws4_request, SignedHeaders=amz-sdk-
invocation-id;amz-sdk-request;host;x-amz-content-sha256;x-amz-date;x-amz-te,
Signature=e367fa0bc217a6a65675bb743e1280cf12f8e8d566196a816d948fdf0b42ca1a}{User-
Agent: aws-sdk-java/2.17.230 Mac_OS_X/12.5 OpenJDK_64-Bit_Server_VM/25.332-b08
Java/1.8.0_332 vendor/Amazon.com_Inc. io/sync http/URLConnection cfg/retry-mode/
legacy}{x-amz-content-sha256: UNSIGNED-PAYLOAD}{X-Amz-Date: 20220927T133955Z}{x-amz-
te: append-md5}{Host: tkhill-test1.s3.amazonaws.com}{Accept: text/html, image/gif,
image/jpeg, *, q=.2, */*; q=.2}{Connection: keep-alive}
<Response> FINE: sun.net.www.MessageHeader@70a36a6611 pairs: {null: HTTP/1.1
200 OK}{x-amz-id-2: sAFeZD0kDUMsBbkjdjyDZw7P0oocb4C9KbiuzfJ6TWKQsGXHM/
dFu0vr2tUb7Y1wEHGdJ3DSIxq0=}{x-amz-request-id: P9QW9SMZ97FKZ9X7}{Date: Tue,
27 Sep 2022 13:39:57 GMT}{Last-Modified: Tue, 13 Sep 2022 14:38:12 GMT}{ETag:
"2cbe5ad4a064cedec33b452bebf48032"}{x-amz-transfer-encoding: append-md5}{Accept-
Ranges: bytes}{Content-Type: text/plain}{Server: AmazonS3}{Content-Length: 67}
```

Per visualizzare i corpi di richiesta/risposta, aggiungili alle proprietà `-Djavax.net.debug=all` JVM. Questa proprietà aggiuntiva registra una grande quantità di informazioni, incluse tutte le informazioni SSL.

Nella console di registro o nel file di registro, cercate "GET" o accedete rapidamente "POST" alla sezione del registro contenente le richieste e le risposte effettive. Cerca "Plaintext before

ENCRYPTION" le richieste e le risposte "Plaintext after DECRYPTION" per visualizzare il testo completo delle intestazioni e dei corpi.

## NettyNioAsyncHttpClient

Se il client di servizio asincrono utilizza l'impostazione predefinita `NettyNioAsyncHttpClient`, aggiungi due logger aggiuntivi al `log4j2.xml` file per registrare le intestazioni HTTP e i corpi di richiesta/risposta.

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

Ecco un esempio completo: `log4j2.xml`

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

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

Queste impostazioni registrano tutti i dettagli dell'intestazione e i corpi di richiesta/risposta.

## AwsCrtAsyncHttpClient

Se il client di servizio è stato configurato per l'utilizzo di un'istanza di `AwsCrtAsyncHttpClient`, è possibile registrare i dettagli impostando le proprietà del sistema JVM o a livello di codice.

Log to a file at "Debug" level

**Utilizzo delle proprietà del sistema:**

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

**A livello di codice:**

```
import software.amazon.awssdk.crt.Log;

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

**Log to the console at "Debug" level****Utilizzo delle proprietà del sistema:**

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

**A livello di codice:**

```
import software.amazon.awssdk.crt.Log;

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

Per motivi di sicurezza, a livello «Trace» `AwsCrtAsyncHttpClient` i log rispondono solo alle intestazioni. Le intestazioni delle richieste, i corpi delle richieste e i corpi delle risposte non vengono registrati.

## Imposta il TTL JVM per le ricerche dei nomi DNS

Java Virtual Machine (JVM) memorizza nella cache le ricerche dei nomi DNS. Quando la JVM risolve un nome host in un indirizzo IP, memorizza l'indirizzo IP nella cache per un periodo di tempo specificato, noto come (TTL). time-to-live

Poiché AWS le risorse utilizzano voci di nomi DNS che cambiano occasionalmente, si consiglia di configurare la JVM con un valore TTL di 5 secondi. Questo garantisce che quando l'indirizzo IP di una risorsa cambia, l'applicazione potrà ricevere e utilizzare il nuovo indirizzo IP della risorsa richiedendo il DNS.

In alcune configurazioni Java, il TTL predefinito di JVM è impostato in modo da non aggiornare mai le voci DNS finché JVM non viene riavviato. Pertanto, se l'indirizzo IP di una AWS risorsa

cambia mentre l'applicazione è ancora in esecuzione, non sarà possibile utilizzare tale risorsa finché non si riavvia manualmente la JVM e le informazioni IP memorizzate nella cache non vengono aggiornate. In questo caso, è fondamentale impostare il valore TTL della JVM in modo che aggiorni periodicamente le informazioni IP memorizzate nella cache.

## Come impostare il TTL JVM

Per modificare il TTL della JVM, imposta il valore della proprietà di sicurezza [networkaddress.cache.ttl](#), imposta la proprietà nel file per Java 8 o nel `networkaddress.cache.ttl` file per Java 11 o versioni successive. `$JAVA_HOME/jre/lib/security/java.security` `$JAVA_HOME/conf/security/java.security`

Quello che segue è un frammento di un file che mostra la cache TTL impostata su 5 secondi.  
`java.security`

```
#
# This is the "master security properties file".
#
# An alternate java.security properties file may be specified
...
# 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
...
```

Tutte le applicazioni eseguite sulla JVM rappresentata dalla variabile di `$JAVA_HOME` ambiente utilizzano questa impostazione.

## Best practice per AWS SDK for Java 2.x

Questa sezione elenca le best practice per l'utilizzo dell'SDK for Java 2.x.

### Argomenti

- [Riutilizza un client SDK, se possibile](#)
- [Chiudi i flussi di input dalle operazioni del client](#)
- [Ottimizza le configurazioni HTTP in base ai test delle prestazioni](#)

- [Usa OpenSSL per il client HTTP basato su Netty](#)
- [Configura i timeout delle API](#)
- [Usa le metriche](#)

## Riutilizza un client SDK, se possibile

Ogni client SDK mantiene il proprio pool di connessioni HTTP. Una connessione già esistente nel pool può essere riutilizzata con una nuova richiesta per ridurre il tempo necessario per stabilire una nuova connessione. Ti consigliamo di condividere una singola istanza del client per evitare il sovraccarico dovuto alla presenza di troppi pool di connessioni che non vengono utilizzati in modo efficace. Tutti i client SDK sono thread-safe.

Se non desideri condividere un'istanza client, richiedi `close()` all'istanza di rilasciare le risorse quando il client non è necessario.

## Chiudi i flussi di input dalle operazioni del client

Per operazioni di streaming [S3Client#getObject](#), ad esempio se lavori [ResponseInputStream](#) direttamente con, ti consigliamo di fare quanto segue:

- Leggi tutti i dati del flusso di input il prima possibile.
- Chiudi il flusso di input il prima possibile.

Forniamo questi consigli perché il flusso di input è un flusso diretto di dati proveniente dalla connessione HTTP e la connessione HTTP sottostante non può essere riutilizzata finché tutti i dati dello stream non sono stati letti e lo stream non viene chiuso. Se queste regole non vengono seguite, il client può esaurire le risorse allocando troppe connessioni HTTP aperte ma inutilizzate.

## Ottimizza le configurazioni HTTP in base ai test delle prestazioni

L'SDK fornisce un set di [configurazioni http predefinite](#) che si applicano a casi d'uso generali. Consigliamo ai clienti di ottimizzare le configurazioni HTTP per le loro applicazioni in base ai loro casi d'uso.

Come buon punto di partenza, l'SDK offre una funzionalità di [configurazione predefinita intelligente](#). Questa funzionalità è disponibile a partire dalla versione 2.17.102. È possibile scegliere una modalità in base al caso d'uso, che fornisce valori di configurazione ragionevoli.

## Usa OpenSSL per il client HTTP basato su Netty

Per impostazione predefinita, l'SDK [NettyNioAsyncHttpClient](#) utilizza l'implementazione SSL predefinita di JDK come `SslProvider`. I nostri test hanno rilevato che OpenSSL offre prestazioni migliori rispetto all'implementazione predefinita di JDK. La community di Netty [consiglia inoltre di utilizzare OpenSSL](#).

Per usare OpenSSL, `netty-tcnative` aggiungilo alle tue dipendenze. Per i dettagli di configurazione, consulta la documentazione del progetto [Netty](#).

Dopo aver `netty-tcnative` configurato il progetto, l'`NettyNioAsyncHttpClient` istanza selezionerà automaticamente OpenSSL. In alternativa, puoi impostarlo in `SslProvider` modo esplicito utilizzando il `NettyNioAsyncHttpClient` builder, come mostrato nel seguente frammento.

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

## Configura i timeout delle API

L'SDK fornisce [valori predefiniti](#) per alcune opzioni di timeout, come il timeout di connessione e i timeout dei socket, ma non per i timeout delle chiamate API o i timeout dei singoli tentativi di chiamata API. È buona norma impostare i timeout sia per i singoli tentativi che per l'intera richiesta. Ciò garantirà un guasto rapido dell'applicazione in modo ottimale in caso di problemi temporanei che potrebbero causare tempi più lunghi per il completamento dei tentativi di richiesta o problemi di rete irreversibili.

È possibile configurare i timeout per tutte le richieste effettuate dai client di un servizio utilizzando and. [ClientOverrideConfiguration#apiCallAttemptTimeout](#)  
[ClientOverrideConfiguration#apiCallTimeout](#)

L'esempio seguente mostra la configurazione di un client Amazon S3 con valori di timeout personalizzati.

```
S3Client.builder()
    .overrideConfiguration(
        b -> b.apiCallTimeout(Duration.ofSeconds(<custom value>)))
```



```
        .apiCallAttemptTimeout(Duration.ofMillis(<custom value>)))  
    .build();
```

## apiCallAttemptTimeout

Questa impostazione imposta la quantità di tempo per un singolo tentativo HTTP, dopodiché la chiamata API può essere ritentata.

## apiCallTimeout

Il valore di questa proprietà configura la quantità di tempo per l'intera esecuzione, inclusi tutti i tentativi di nuovo tentativo.

In alternativa all'impostazione di questi valori di timeout sul client del servizio, è possibile utilizzare [RequestOverrideConfiguration#apiCallTimeout\(\)](#) e [RequestOverrideConfiguration#apiCallAttemptTimeout\(\)](#) configurare una singola richiesta.

L'esempio seguente configura una singola `listBuckets` richiesta con valori di timeout personalizzati.

```
s3Client.listBuckets(lbr -> lbr.overrideConfiguration(  
    b -> b.apiCallTimeout(Duration.ofSeconds(<custom value>))  
        .apiCallAttemptTimeout(Duration.ofMillis(<custom value>))));
```

Quando si utilizzano queste proprietà insieme, si imposta un limite rigido al tempo totale impiegato per tutti i tentativi tra più tentativi. È inoltre possibile impostare una richiesta HTTP individuale in modo che abbia esito negativo rapidamente in caso di richiesta lenta.

## Usa le metriche

L'SDK for Java [può raccogliere](#) metriche per i client di servizio dell'applicazione. Puoi utilizzare queste metriche per identificare problemi di prestazioni, esaminare le tendenze complessive di utilizzo, esaminare le eccezioni restituite dai client di servizio o per approfondire un problema particolare.

Ti consigliamo di raccogliere parametri, quindi analizzare Amazon CloudWatch Logs, per ottenere una comprensione più approfondita delle prestazioni della tua applicazione.

## Domande frequenti per la soluzione dei problemi

Durante l'utilizzo di AWS SDK for Java 2.x nelle applicazioni, è possibile che si verifichino gli errori di runtime elencati in questo argomento. Utilizzate i suggerimenti riportati di seguito per individuare la causa principale e risolvere l'errore.

### Come posso correggere l'errore "**java.net.SocketException: ripristino della connessione**» o «il server non è riuscito a completare la risposta»?

Un errore di reimpostazione della connessione indica che l'host Servizio AWS, l'intermediario (ad esempio un gateway NAT, un proxy, un sistema di bilanciamento del carico) ha chiuso la connessione prima del completamento della richiesta. Poiché le cause sono molteplici, per trovare una soluzione è necessario conoscere il motivo per cui la connessione viene chiusa. I seguenti elementi causano in genere la chiusura di una connessione.

- La connessione non è attiva. Ciò è comune nelle operazioni di streaming, in cui i dati non vengono scritti sul cavo o dal cavo per un certo periodo di tempo, quindi una parte intermediaria rileva che la connessione è interrotta e la chiude. Per evitare che ciò accada, assicurati che l'applicazione scarichi o carichi attivamente i dati.
- Hai chiuso il client HTTP o SDK. Assicurati di non chiudere le risorse mentre sono in uso.
- Un proxy configurato in modo errato. Prova a bypassare tutti i proxy che hai configurato per vedere se il problema viene risolto. Se questo risolve il problema, il proxy sta chiudendo la connessione per qualche motivo. Cerca il tuo proxy specifico per determinare il motivo per cui sta chiudendo la connessione.

Se non riesci a identificare il problema, prova a eseguire un dump TCP per una connessione interessata sul lato client della rete (ad esempio, dopo tutti i proxy che controlli).

Se vedi che l'AWS endpoint sta inviando un TCP RST (ripristino), [contatta il servizio interessato](#) per vedere se è in grado di determinare il motivo del ripristino. Preparati a fornire gli ID delle richieste e gli orari in cui si è verificato il problema. Il team di AWS supporto potrebbe anche trarre vantaggio dai [registri telefonici](#) che mostrano esattamente quali byte l'applicazione invia e riceve e quando.

### Come posso correggere il «timeout della connessione»?

Un errore di timeout della connessione indica che l'host Servizio AWS, il server o qualsiasi intermediario (ad esempio un gateway NAT, un proxy, un sistema di bilanciamento del carico) non è

riuscito a stabilire una nuova connessione con il server entro il timeout di connessione configurato. I seguenti elementi descrivono le cause più comuni di questo problema.

- Il timeout di connessione configurato è troppo basso. Per impostazione predefinita, il timeout della connessione è di 2 secondi in AWS SDK for Java 2.x. Se il timeout di connessione è impostato su un valore troppo basso, è possibile che venga visualizzato questo errore. Il timeout di connessione consigliato è di 1 secondo se si effettuano solo chiamate interregionali e di 3 secondi se si effettuano richieste interregionali.
- Un proxy non configurato correttamente. Prova a bypassare tutti i proxy che hai configurato per vedere se il problema viene risolto. Se questo risolve il problema, il motivo del timeout della connessione è il proxy. Cerca il tuo proxy specifico per determinare il motivo per cui ciò sta accadendo.

Se non riesci a identificare il problema, prova a eseguire un dump TCP per una connessione interessata sul lato client della rete (ad esempio, dopo tutti i proxy che controlli) per esaminare eventuali problemi di rete.

## Come posso risolvere il problema

### "`java.net.SocketTimeoutException: Read timeout`»?

Un errore di timeout di lettura indica che la JVM ha tentato di leggere i dati dal sistema operativo sottostante, ma i dati non sono stati restituiti entro il tempo configurato tramite l'SDK. Questo errore può verificarsi se il sistema operativo, la o qualsiasi parte intermedia (ad esempio Servizio AWS, un gateway NAT, un proxy, un sistema di bilanciamento del carico) non riesce a inviare i dati entro il tempo previsto dalla JVM. Poiché le cause sono molteplici, per trovare una soluzione è necessario conoscere il motivo per cui i dati non vengono restituiti.

Prova a eseguire un dump TCP per una connessione interessata nella periferia client della rete (ad esempio, dopo tutti i proxy che controlli).

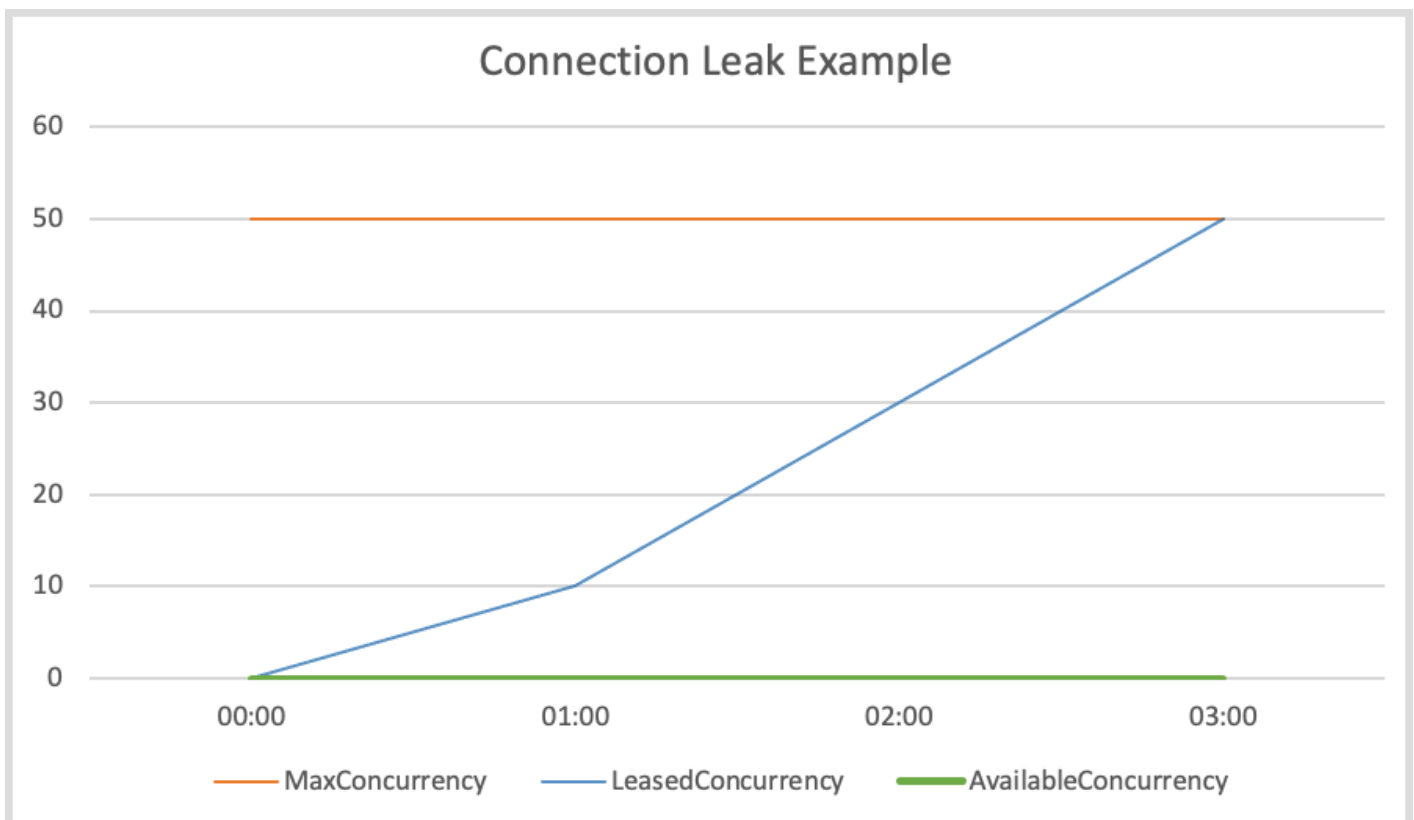
[Se vedi che l' AWS endpoint sta inviando un TCP RST \(reset\), contatta il servizio interessato.](#)

Preparati a fornire gli ID delle richieste e gli orari in cui si è verificato il problema. Il team di AWS supporto potrebbe anche trarre vantaggio dai [registri telefonici](#) che mostrano esattamente quali byte l'applicazione invia e riceve e quando.

## Come posso correggere l'errore «Impossibile eseguire la richiesta HTTP: timeout in attesa di connessione dal pool»?

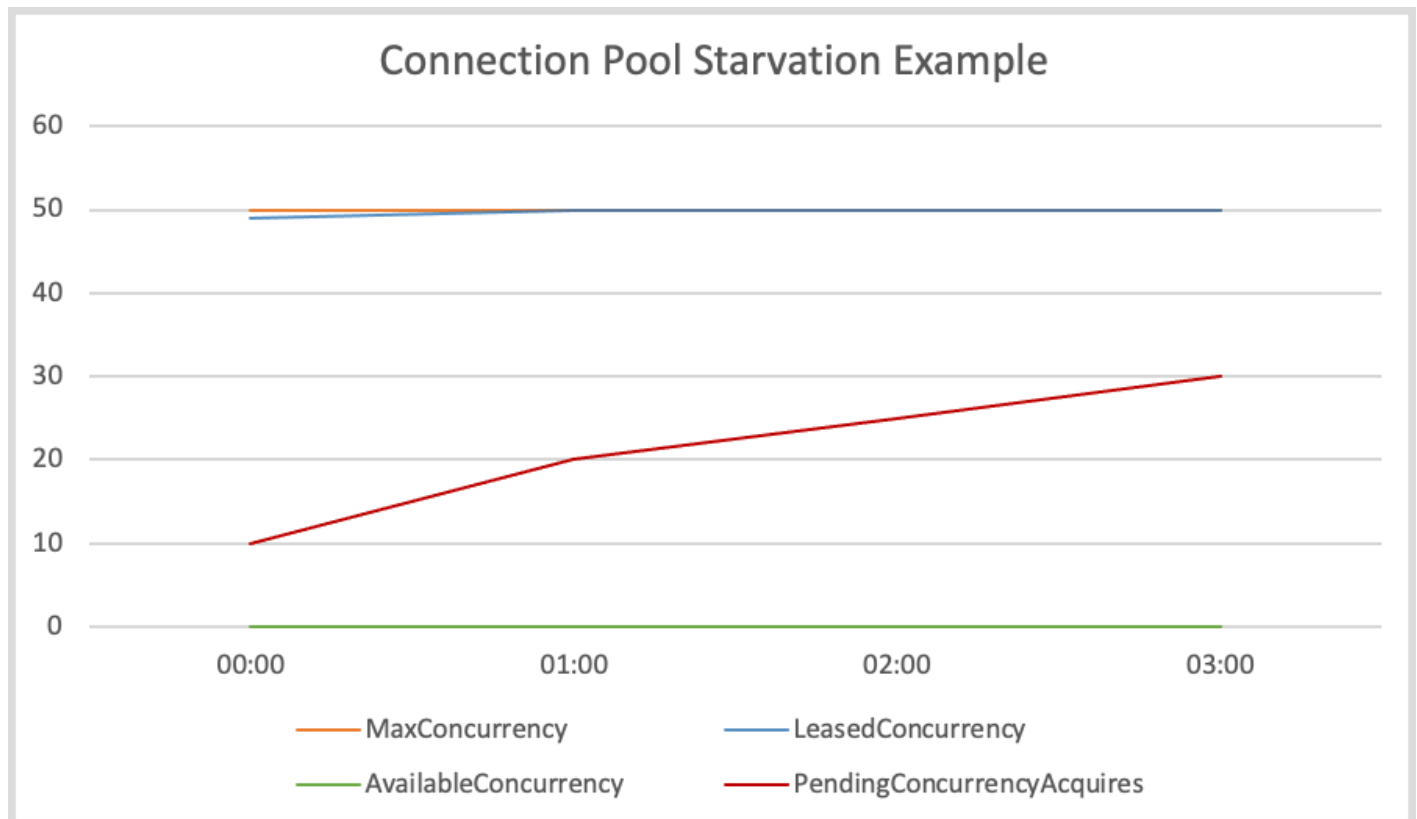
Questo errore indica che una richiesta non può ottenere una connessione dal pool entro il tempo massimo specificato. Per risolvere il problema, ti consigliamo di [abilitare i parametri SDK lato client per pubblicare i parametri su Amazon](#). CloudWatch Le metriche HTTP possono aiutare a restringere la causa principale. I seguenti elementi descrivono le cause più comuni di questo errore.

- Perdita di connessione. Puoi indagare su questo problema controllando LeasedConcurrencyAvailableConcurrency, e MaxConcurrency metriche. Se LeasedConcurrency aumenta fino a raggiungere l'MaxConcurrencyobiettivo ma non diminuisce mai, potrebbe esserci una perdita di connessione. Una causa comune di una perdita di dati è che un'operazione di streaming, ad esempio un metodo S3, non viene chiusa. getObject [Consigliamo all'applicazione di leggere tutti i dati dal flusso di input il prima possibile e di chiudere il flusso di input in seguito](#). La tabella seguente mostra come potrebbero apparire le metriche SDK per le perdite di connessione.



- Interruzione del pool di connessioni. Ciò può verificarsi se la frequenza delle richieste è troppo elevata e la dimensione del pool di connessioni configurato non è in grado di soddisfare la richiesta. La dimensione predefinita del pool di connessioni è 50 e quando le connessioni nel pool

raggiungono il massimo, il client HTTP mette in coda le richieste in arrivo finché le connessioni non diventano disponibili. Il grafico seguente mostra come potrebbero apparire le metriche SDK in caso di carenza di pool di connessioni.



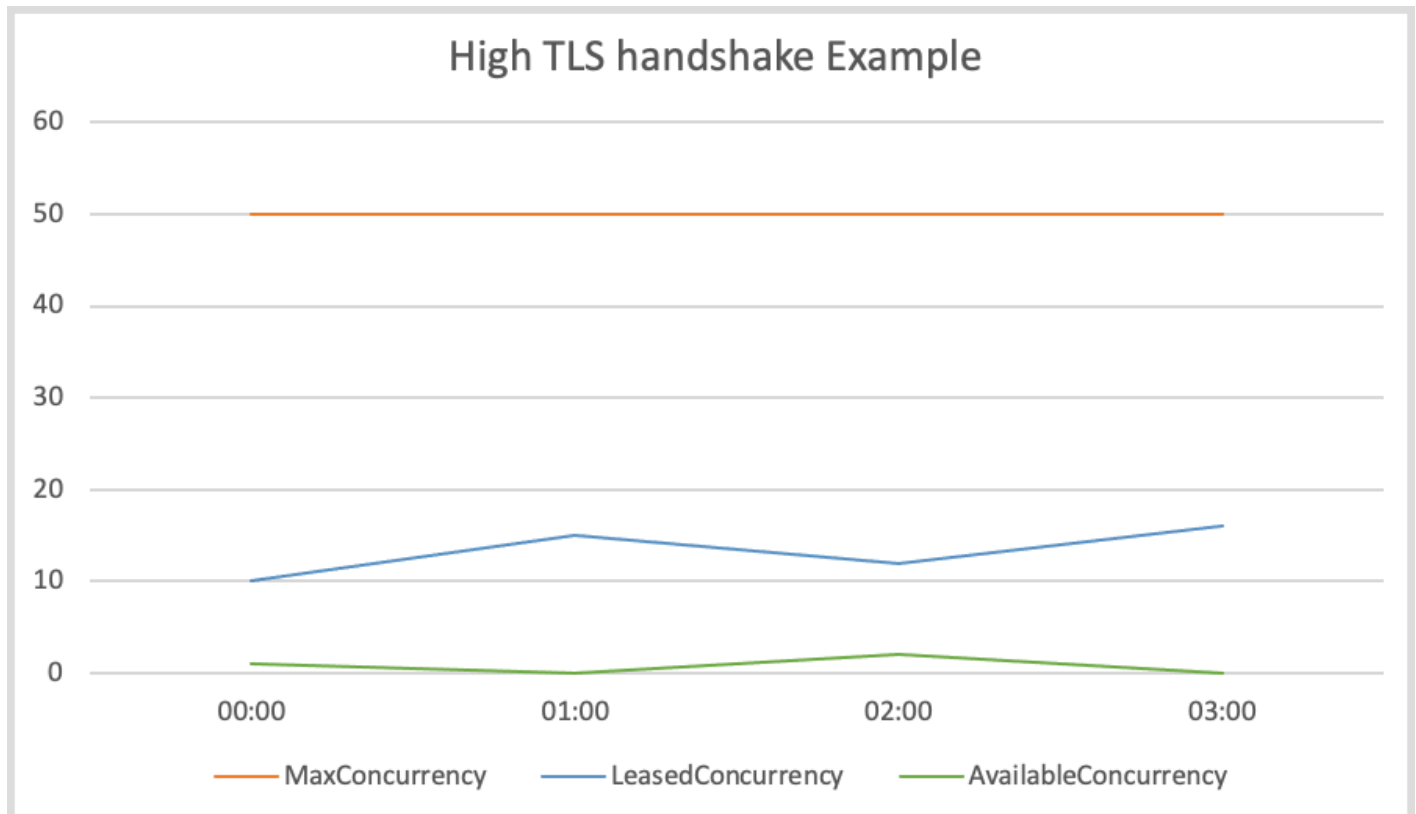
Per mitigare questo problema, prendi in considerazione l'idea di intraprendere una delle seguenti azioni.

- Aumentare la dimensione del pool di connessioni,
- Aumentare il timeout di acquisizione.
- Rallenta la frequenza delle richieste.

Aumentando il numero massimo di connessioni, è possibile aumentare la velocità effettiva del client (a meno che l'interfaccia di rete non sia già completamente utilizzata). Tuttavia, alla fine è possibile raggiungere le limitazioni del sistema operativo relative al numero di descrittori di file utilizzati dal processo. Se utilizzi già completamente l'interfaccia di rete o non riesci ad aumentare ulteriormente il numero di connessioni, prova ad aumentare il timeout di acquisizione. Con l'aumento, avrai più tempo a disposizione per le richieste di acquisizione di una connessione prima del timeout. Se le connessioni non si liberano, le richieste successive continueranno a scadere.

Se non riesci a risolvere il problema utilizzando i primi due meccanismi, rallenta la frequenza delle richieste provando le seguenti opzioni.

- Semplifica le tue richieste in modo che i grandi picchi di traffico non sovraccarichino il client.
- Sii più efficiente con le chiamate a Servizi AWS
- Aumenta il numero di host che inviano richieste.
- I thread di I/O sono troppo occupati. Questo vale solo se si utilizza un client SDK asincrono con [NettyNioAsyncHttpClient](#). Se la `AvailableConcurrency` metrica non è bassa, ovvero indica che le connessioni sono disponibili nel pool, ma `ConcurrencyAcquireDuration` è elevata, è possibile che i thread di I/O non siano in grado di gestire le richieste. Assicurati di non passare `Runnable::run` per un [futuro esecutore di completamento](#) e di non eseguire attività dispendiose in termini di tempo nella catena di completamento delle risposte future, poiché ciò può bloccare un thread di I/O. In caso contrario, valuta la possibilità di aumentare il numero di thread di I/O utilizzando il metodo [eventLoopGroupBuilder](#). Per riferimento, il numero predefinito di thread di I/O per un'istanza di `NettyNioAsyncHttpClient` è il doppio del numero di core CPU dell'host.
- Latenza di handshake TLS elevata. Se la `AvailableConcurrency` metrica è vicina a 0 ed `LeasedConcurrency` è inferiore a `MaxConcurrency`, è possibile che la latenza dell'handshake TLS sia elevata. Il grafico seguente mostra come potrebbero apparire le metriche SDK per un'elevata latenza di handshake TLS.



[Per i client HTTP offerti da Java SDK che non sono basati su CRT, provate ad abilitare i log TLS per risolvere i problemi TLS.](#) [Per il client HTTP basato su CRT, prova ad AWS abilitare i log CRT.AWS](#) [Se notate che l' AWS endpoint sembra impiegare molto tempo per eseguire un handshake TLS, contattate il servizio interessato.](#)

## Come posso riparare un `NoClassDefFoundError`, or? `NoSuchMethodError` `NoSuchFieldError`

A `NoClassDefFoundError` indica che non è stato possibile caricare una classe in fase di esecuzione. Le due cause più comuni di questo errore sono:

- la classe non esiste nel classpath perché nel classpath manca il JAR o nel classpath è presente una versione errata del JAR.
- la classe non è stata caricata perché il suo iniziatore statico ha generato un'eccezione.

Allo stesso modo, `NoSuchMethodError` s e `NoSuchFieldError` s in genere derivano da una versione JAR non corrispondente. Si consiglia di eseguire le seguenti operazioni.

1. Controlla le tue dipendenze per assicurarti di utilizzare la stessa versione di tutti i jar SDK. Il motivo più comune per cui non è possibile trovare una classe, un metodo o un campo è quando esegui l'aggiornamento a una nuova versione del client ma continui a utilizzare una vecchia versione di dipendenza SDK «condivisa». La nuova versione del client potrebbe tentare di utilizzare classi che esistono solo nelle dipendenze SDK «condivise» più recenti. Prova a eseguire `mvn dependency:tree` o `gradle dependencies` (per Gradle) a verificare che tutte le versioni della libreria SDK corrispondano. Per evitare completamente questo problema in futuro, consigliamo di utilizzare [BOM \(Bill of Materials\)](#) per gestire le versioni dei moduli SDK.

L'esempio seguente mostra un esempio di versioni SDK miste.

```
[INFO] +- software.amazon.awssdk:dynamodb:jar:2.20.00:compile
[INFO] |   +- software.amazon.awssdk:aws-core:jar:2.13.19:compile
[INFO] +- software.amazon.awssdk:netty-nio-client:jar:2.20.00:compile
```

La versione di `dynamodb` è 2.20.00 e la versione di `aws-core` è 2.13.19. Anche la versione degli `aws-core` artefatti dovrebbe essere la 2.20.00.

2. Controlla le istruzioni nelle prime fasi dei log per vedere se una classe non riesce a caricarsi a causa di un errore di inizializzazione statico. La prima volta che la classe non riesce a caricarsi, potrebbe generare un'eccezione diversa e più utile che specifica il motivo per cui la classe non può essere caricata. Questa eccezione potenzialmente utile si verifica solo una volta, quindi le istruzioni di registro successive segnaleranno solo che la classe non è stata trovata.
3. Controllate il processo di distribuzione per assicurarvi che distribuisca effettivamente i file JAR richiesti insieme all'applicazione. È possibile che tu stia compilando con la versione corretta, ma il processo che crea il classpath per l'applicazione esclude una dipendenza richiesta.

Come posso correggere un errore "**SignatureDoesNotMatch**" o l'errore «La firma della richiesta che abbiamo calcolato non corrisponde alla firma che hai fornito»?

Un `SignatureDoesNotMatch` errore indica che la firma generata da AWS SDK for Java e la firma generata da Servizio AWS non corrispondono. I seguenti elementi descrivono le cause potenziali.

- Un delegato o un intermediario modifica la richiesta. Ad esempio, un proxy o un load balancer potrebbe modificare un'intestazione, un percorso o una stringa di query firmata dall'SDK.



- Il servizio e l'SDK differiscono nel modo in cui codificano la richiesta quando ciascuno genera la stringa da firmare.

Per eseguire il debug di questo problema, ti consigliamo di [abilitare la registrazione di debug](#) per l'SDK. Prova a riprodurre l'errore e trova la richiesta canonica generata dall'SDK. Nel registro, la richiesta canonica è etichettata con `AWS4 Canonical Request: ...` e la stringa da firmare è etichettata `AWS4 String to sign: ...`.

Se non puoi abilitare il debug, ad esempio perché è riproducibile solo in produzione, aggiungi una logica all'applicazione che registra le informazioni sulla richiesta quando si verifica l'errore. È quindi possibile utilizzare tali informazioni per provare a replicare l'errore al di fuori della produzione in un test di integrazione con la registrazione di debug abilitata.

Dopo aver raccolto la richiesta canonica e la stringa da firmare, confrontale con la [specifica AWS Signature Version 4](#) per determinare se ci sono problemi nel modo in cui l'SDK ha generato la stringa da firmare. Se qualcosa sembra sbagliato, puoi creare una [segnalazione di GitHub bug](#) a. AWS SDK for Java

Se non appare nulla di sbagliato, puoi confrontare la stringa dell'SDK da firmare con la stringa per firmare che alcuni Servizi AWS restituiscono come parte della risposta all'errore (Amazon S3, ad esempio). Se non è disponibile, [contatta il servizio interessato](#) per vedere quale richiesta canonica e stringa da firmare hanno generato per il confronto. Questi confronti possono aiutare a identificare gli intermediari che potrebbero aver modificato la richiesta o le differenze di codifica tra il servizio e il client.

Per ulteriori informazioni di base sulla firma delle richieste, consulta [Firmare le richieste AWS API nella Guida](#) per l' AWS Identity and Access Management utente.

Example di una richiesta canonica

```
PUT
/Example-Bucket/Example-Object
partNumber=19&uploadId=string
amz-sdk-invocation-id:f8c2799d-367c-f024-e8fa-6ad6d0a1afb9
amz-sdk-request:attempt=1; max=4
content-encoding:aws-chunked
content-length:51
content-type:application/octet-stream
host:xxxxx
x-amz-content-sha256:STREAMING-UNSIGNED-PAYLOAD-TRAILER
```

```
x-amz-date:20240308T034733Z
x-amz-decoded-content-length:10
x-amz-sdk-checksum-algorithm: CRC32
x-amz-trailer:x-amz-checksum-crc32
```

### Example di una stringa da firmare

```
AWS4-HMAC-SHA256
20240308T034435Z
20240308/us-east-1/s3/aws4_request
5f20a7604b1ef65dd89c333fd66736fdef9578d11a4f5d22d289597c387dc713
```

## Come posso correggere l'errore "**java.lang.IllegalStateException: Connection pool shut down**"?

Questo errore indica che il pool di connessioni HTTP Apache sottostante è stato chiuso. I seguenti elementi descrivono le cause potenziali.

- Il client SDK è stato chiuso prematuramente. L'SDK chiude il pool di connessioni solo quando il client associato viene chiuso. Assicurati di non chiudere le risorse mentre sono in uso.
- È **java.lang.Error** stato lanciato A. Errori come quello che `OutOfMemoryError` causa la [chiusura](#) di un pool di connessioni HTTP di Apache. Esamina i log per individuare eventuali tracce dello stack di errori. Esamina anche il codice per individuare i punti in cui rileva `Throwable` o `Error` ma ingoia l'output che impedisce la comparsa dell'errore. Se il codice non riporta errori, riscrivilo in modo che le informazioni vengano registrate. Le informazioni registrate aiutano a determinare la causa principale dell'errore.
- Hai tentato di utilizzare il provider di credenziali restituito **`DefaultCredentialsProvider#create()`** dopo la sua chiusura. [`DefaultCredentialsProvider#create`](#) restituisce un'istanza singleton, quindi se è chiusa e il codice chiama il `resolveCredentials` metodo, l'eccezione viene generata dopo la scadenza delle credenziali (o token) memorizzate nella cache.

Controlla il codice per individuare i punti in cui `DefaultCredentialsProvider` è chiuso, come illustrato negli esempi seguenti.

- L'istanza singleton viene chiusa chiamando `DefaultCredentialsProvider#close()`.

```
DefaultCredentialsProvider defaultCredentialsProvider =
    DefaultCredentialsProvider.create(); // Singleton instance returned.
```

```

AwsCredentials credentials = defaultCredentialsProvider.resolveCredentials();

// Make calls to Servizi AWS.

defaultCredentialsProvider.close(); // Explicit close.

// Make calls to Servizi AWS.

// After the credentials expire, either of the following calls eventually results
// in a "Connection pool shut down" exception.
credentials = defaultCredentialsProvider.resolveCredentials();
// Or
credentials = DefaultCredentialsProvider.create().resolveCredentials();

```

- Invoca `DefaultCredentialsProvider#create()` in un blocco `try-with-resources`

```

try (DefaultCredentialsProvider defaultCredentialsProvider =
    DefaultCredentialsProvider.create()) {
    AwsCredentials credentials = defaultCredentialsProvider.resolveCredentials();

    // Make calls to Servizi AWS.

} // After the try-with-resources block exits, the singleton
// DefaultCredentialsProvider is closed.

// Make calls to Servizi AWS.

DefaultCredentialsProvider defaultCredentialsProvider =
    DefaultCredentialsProvider.create(); // The closed singleton instance is returned.
// If the credentials (or token) has expired, the following call results in the
// error.
AwsCredentials credentials = defaultCredentialsProvider.resolveCredentials();

```

Crea una nuova istanza non singleton chiamando

`DefaultCredentialsProvider.builder().build()` se il tuo codice ha chiuso l'istanza singleton e devi risolvere le credenziali utilizzando un `DefaultCredentialsProvider`

# Utilizza le funzionalità della versione 2.x AWS SDK for Java

## Caratteristiche generali

L'SDK for Java 2.x contiene diverse funzionalità che semplificano la programmazione Servizi AWS .

- [L'SDK nasconde i complessi meccanismi alla base del recupero dei risultati impaginati e del polling delle risorse.](#)
- La [programmazione asincrona con I/O non bloccante consente di scrivere codice simultaneo con prestazioni migliori.](#) L'SDK offre i vantaggi di [HTTP/2](#), come la latenza ridotta, ove possibile.
- L'SDK Java può generare [metriche](#) per aiutarti a monitorare lo stato operativo delle tue applicazioni.

## Funzionalità specifiche del servizio

Oltre alle funzionalità generali menzionate in precedenza, Java SDK offre funzionalità specifiche. Servizi AWS

- Amazon S3: per [semplificare il lavoro con file e directory con](#) Amazon S3, l'SDK fornisce S3 Transfer Manager. Per [migliorare le prestazioni e l'affidabilità](#) utilizzando l'API S3 asincrona standard dell'SDK, l'SDK offre il client S3 basato su CRT. AWS
- DynamoDB — [La funzionalità di mappatura orientata agli oggetti è fornita dall'API](#) DynamoDB Enhanced Client. [Lavora con dati in stile JSON e orientati ai documenti utilizzando l'API Enhanced Document.](#)
- IAM: l'API IAM Policy Builder offre un modo [sicuro per i tipi e orientato](#) agli oggetti per creare policy IAM.

## Lavora con risultati impaginati utilizzando la versione 2.x AWS SDK for Java

Molte AWS operazioni restituiscono risultati impaginati quando l'oggetto di risposta è troppo grande per essere restituito in un'unica risposta. Nella AWS SDK for Java versione 1.0, la risposta contiene un token da utilizzare per recuperare la pagina successiva dei risultati. Al contrario, la versione AWS SDK for Java 2.x dispone di metodi di autopaginazione che effettuano più chiamate di servizio per

visualizzare automaticamente la pagina successiva dei risultati. Occorre solo scrivere il codice che elabora i risultati. L'autopaginazione è disponibile sia per i client sincroni che asincroni.

### Note

Questi frammenti di codice presuppongono che tu conosca le basi dell'utilizzo dell'SDK e che tu abbia configurato il tuo ambiente con l'accesso Single Sign-On.

## Impaginazione sincrona

Gli esempi seguenti mostrano i metodi di impaginazione sincroni per elencare gli oggetti in un bucket. Amazon S3

### Iterate sulle pagine

Il primo esempio dimostra l'uso di un oggetto `ListRes` paginator, un'[ListObjectsV2Iterable](#) istanza, per scorrere tutte le pagine di risposta con il metodo `stream`. Il codice scorre sulle pagine di risposta, converte il flusso di risposta in un flusso di [S3Object](#) contenuti e quindi elabora il contenuto dell'oggetto. Amazon S3

Le seguenti importazioni si applicano a tutti gli esempi di questa sezione di impaginazione sincrona.

### Importazioni

```
import java.io.IOException;
import java.nio.ByteBuffer;
import java.util.Random;
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.paginators.ListObjectsV2Iterable;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Request;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Response;
import software.amazon.awssdk.services.s3.model.S3Object;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.DeleteObjectRequest;
import software.amazon.awssdk.services.s3.model.DeleteBucketRequest;
```

```
import software.amazon.awssdk.services.s3.model.CreateMultipartUploadRequest;
import software.amazon.awssdk.services.s3.model.CreateMultipartUploadResponse;
import software.amazon.awssdk.services.s3.model.CompletedMultipartUpload;
import software.amazon.awssdk.services.s3.model.CreateBucketRequest;
import software.amazon.awssdk.services.s3.model.CompletedPart;
import software.amazon.awssdk.services.s3.model.CreateBucketConfiguration;
import software.amazon.awssdk.services.s3.model.UploadPartRequest;
import software.amazon.awssdk.services.s3.model.CompleteMultipartUploadRequest;
import software.amazon.awssdk.services.s3.waiters.S3Waiter;
import software.amazon.awssdk.services.s3.model.HeadBucketRequest;
import software.amazon.awssdk.services.s3.model.HeadBucketResponse;
```

```
ListObjectsV2Request listReq = ListObjectsV2Request.builder()
    .bucket(bucketName)
    .maxKeys(1)
    .build();

ListObjectsV2Iterable listRes = s3.listObjectsV2Paginator(listReq);
// Process response pages
listRes.stream()
    .flatMap(r -> r.contents().stream())
    .forEach(content -> System.out
        .println(" Key: " + content.key() + "
size = " + content.size()));
```

Vedi l'[esempio completo](#) su GitHub

## Iterate sugli oggetti

Negli esempi seguenti vengono mostrati dei metodi per effettuare l'iterazione sugli oggetti restituiti nella risposta invece che delle pagine della risposta. Il `contents` metodo di `ListObjectsV2Iterable` classe restituisce un [SdkIterable](#) che fornisce diversi metodi per elaborare gli elementi di contenuto sottostanti.

Usa uno stream

Il seguente frammento utilizza il `stream` metodo sul contenuto della risposta per iterare sulla raccolta di elementi impaginati.

```
// Helper method to work with paginated collection of items directly.
listRes.contents().stream()
```

```
        .forEach(content -> System.out
                    .println(" Key: " + content.key() + "
size = " + content.size())));
```

[Vedi l'esempio completo su GitHub](#)

Usa un ciclo for-each

Poiché `SdkIterable` estende l'`Iterable` interfaccia, puoi elaborare i contenuti come qualsiasi altro. `Iterable` Il seguente frammento utilizza un `for-each` ciclo standard per scorrere il contenuto della risposta.

```
        for (S3Object content : listRes.contents()) {
            System.out.println(" Key: " + content.key() + " size = " +
content.size());
        }
```

Vedi l'esempio [completo](#) su. [GitHub](#)

## Impaginazione manuale

Se il caso d'uso lo richiede, la paginazione manuale è ancora disponibile. Utilizza il token successivo dell'oggetto di risposta per le richieste successive. L'esempio seguente utilizza un `while` ciclo.

```
        ListObjectsV2Request listObjectsReqManual =
ListObjectsV2Request.builder()
                        .bucket(bucketName)
                        .maxKeys(1)
                        .build();

        boolean done = false;
        while (!done) {
            ListObjectsV2Response listObjResponse =
s3.listObjectsV2(listObjectsReqManual);
            for (S3Object content : listObjResponse.contents()) {
                System.out.println(content.key());
            }

            if (listObjResponse.nextContinuationToken() == null) {
                done = true;
            }
        }
```

```
listObjectsReqManual = listObjectsReqManual.toBuilder()
    .continuationToken(listObjResponse.nextContinuationToken())
    .build();
}
```

Vedi l'[esempio completo](#) su GitHub.

## Impaginazione asincrona

Gli esempi seguenti mostrano i metodi di impaginazione asincrona per elencare le tabelle. DynamoDB

### Iterate su pagine di nomi di tabelle

I due esempi seguenti utilizzano un client DynamoDB asincrono che chiama `ListTablesPaginator` il metodo con una richiesta per ottenere un [ListTablesPublisher](#). `ListTablesPublisher` implementa due interfacce, che forniscono molte opzioni per elaborare le risposte. Esamineremo i metodi di ciascuna interfaccia.

### Usa un **Subscriber**

Il seguente esempio di codice dimostra come elaborare i risultati impaginati utilizzando l'org.reactivestreams.Publisher interfaccia implementata da `ListTablesPublisher`. [Per ulteriori informazioni sul modello Reactive Streams, consulta il repository Reactive Streams. GitHub](#)

Le seguenti importazioni si applicano a tutti gli esempi di questa sezione di impaginazione asincrona.

### Importazioni

```
import io.reactivex.rxjava3.core.Flowable;
import org.reactivestreams.Subscriber;
import org.reactivestreams.Subscription;
import reactor.core.publisher.Flux;
import software.amazon.awssdk.core.async.SdkPublisher;
import software.amazon.awssdk.services.dynamodb.DynamoDbAsyncClient;
import software.amazon.awssdk.services.dynamodb.model.ListTablesRequest;
import software.amazon.awssdk.services.dynamodb.model.ListTablesResponse;
import software.amazon.awssdk.services.dynamodb.paginator.ListTablesPublisher;

import java.util.List;
```



```
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.ExecutionException;
```

Il codice seguente acquisisce un'istanza. `ListTablesPublisher`

```
// Creates a default client with credentials and region loaded from the
// environment.
final DynamoDbAsyncClient asyncClient = DynamoDbAsyncClient.create();

ListTablesRequest listTablesRequest =
ListTablesRequest.builder().limit(3).build();
ListTablesPublisher publisher =
asyncClient.listTablesPaginator(listTablesRequest);
```

Il codice seguente utilizza un'implementazione anonima di `org.reactivestreams.Subscriber` per elaborare i risultati di ogni pagina.

Il metodo `onSubscribe` chiama il metodo `Subscription.request` per avviare le richieste per i dati del publisher. Questo metodo deve essere chiamato per iniziare a ottenere i dati dal publisher.

Il `onNext` metodo del sottoscrittore elabora una pagina di risposta accedendo a tutti i nomi delle tabelle e stampandoli ciascuno. Dopo l'elaborazione della pagina, viene richiesta un'altra pagina all'editore. Questo metodo viene chiamato ripetutamente fino al recupero di tutte le pagine.

Il metodo `onError` viene attivato se si verifica un errore durante il recupero dei dati. Infine, il metodo `onComplete` viene chiamato quando tutte le pagine sono state richieste.

```
// A Subscription represents a one-to-one life-cycle of a Subscriber
subscribing
// to a Publisher.
publisher.subscribe(new Subscriber<ListTablesResponse>() {
    // Maintain a reference to the subscription object, which is required to
request
    // data from the publisher.
    private Subscription subscription;

    @Override
    public void onSubscribe(Subscription s) {
        subscription = s;
        // Request method should be called to demand data. Here we request a
single
        // page.
```

```

        subscription.request(1);
    }

    @Override
    public void onNext(ListTablesResponse response) {
        response.tableNames().forEach(System.out::println);
        // After you process the current page, call the request method to
signal that
        // you are ready for next page.
        subscription.request(1);
    }

    @Override
    public void onError(Throwable t) {
        // Called when an error has occurred while processing the requests.
    }

    @Override
    public void onComplete() {
        // This indicates all the results are delivered and there are no more
pages
        // left.
    }
});

```

Vedi l'[esempio completo](#) su. GitHub

## Usa un **Consumer**

L'`SdkPublisher` interfaccia che `ListTablesPublisher` implementa ha un `subscribe` metodo che accetta un `Consumer` e restituisce un `CompletableFuture<Void>`.

Il `subscribe` metodo di questa interfaccia può essere utilizzato per casi d'uso semplici in cui un `org.reactivestreams.Subscriber` potrebbe comportare un sovraccarico eccessivo. Poiché il codice seguente utilizza ogni pagina, chiama il `tableNames` metodo su ciascuna di esse. Il `tableNames` metodo restituisce uno `java.util.List` dei nomi di tabella DynamoDB elaborati con il metodo. `forEach`

```

// Use a Consumer for simple use cases.
CompletableFuture<Void> future = publisher.subscribe(
    response -> response.tableNames()
        .forEach(System.out::println));

```

Vedi l'[esempio completo](#) su GitHub

## Esegui iterazioni sui nomi delle tabelle

Negli esempi seguenti vengono mostrati dei metodi per effettuare l'iterazione sugli oggetti restituiti nella risposta invece che delle pagine della risposta. Analogamente all'esempio sincrono di Amazon S3 illustrato in precedenza con il `contents` relativo metodo, la classe `ListTablesPublisher` di risultati asincrona di DynamoDB ha il metodo pratico per interagire con `tableNames` la raccolta di elementi sottostante. Il tipo di restituzione del `tableNames` metodo è un metodo [SdkPublisher](#) che può essere utilizzato per richiedere elementi su tutte le pagine.

### Usa un **Subscriber**

Il codice seguente acquisisce uno `SdkPublisher` dei nomi di tabella sottostanti.

```
// Create a default client with credentials and region loaded from the
// environment.
final DynamoDbAsyncClient asyncClient = DynamoDbAsyncClient.create();

ListTablesRequest listTablesRequest =
ListTablesRequest.builder().limit(3).build();
ListTablesPublisher listTablesPublisher =
asyncClient.listTablesPaginator(listTablesRequest);
SdkPublisher<String> publisher = listTablesPublisher.tableNames();
```

Il codice seguente utilizza un'implementazione anonima di `org.reactivestreams.Subscriber` per elaborare i risultati di ogni pagina.

Il `onNext` metodo del sottoscrittore elabora un singolo elemento della raccolta. In questo caso, è un nome di tabella. Dopo l'elaborazione del nome della tabella, viene richiesto un altro nome di tabella all'editore. Questo metodo viene chiamato ripetutamente fino al recupero di tutti i nomi delle tabelle.

```
// Use a Subscriber.
publisher.subscribe(new Subscriber<String>() {
    private Subscription subscription;

    @Override
    public void onSubscribe(Subscription s) {
        subscription = s;
        subscription.request(1);
    }
});
```

```
        @Override
        public void onNext(String tableName) {
            System.out.println(tableName);
            subscription.request(1);
        }

        @Override
        public void onError(Throwable t) {
        }

        @Override
        public void onComplete() {
        }
    });
```

Vedi l'[esempio completo](#) su GitHub

## Usa un **Consumer**

L'esempio seguente utilizza il `subscribe` metodo `SdkPublisher` che richiede `Consumer` a per elaborare ogni elemento.

```
// Use a Consumer.
CompletableFuture<Void> future = publisher.subscribe(System.out::println);
future.get();
```

Vedi l'[esempio completo](#) su GitHub.

## Usa una libreria di terze parti

Puoi utilizzare altre librerie di terze parti anziché implementare un sottoscrittore personalizzato. Questo esempio dimostra l'uso di RxJava, ma è possibile utilizzare qualsiasi libreria che implementa le interfacce di flusso reattivo. Vedi la [pagina RxJava wiki su GitHub per ulteriori informazioni su quella libreria](#).

Per utilizzare la libreria, aggiungerla come una dipendenza. Se utilizzi Maven, l'esempio illustra lo snippet POM da utilizzare.

### Voce POM

```
<dependency>
```

```
<groupId>io.reactivex.rxjava3</groupId>
<artifactId>rxjava</artifactId>
<version>3.1.6</version>
</dependency>
```

## Codice

```
DynamoDbAsyncClient asyncClient = DynamoDbAsyncClient.create();
ListTablesPublisher publisher =
asyncClient.listTablesPaginator(ListTablesRequest.builder()
    .build());

// The Flowable class has many helper methods that work with
// an implementation of an org.reactivestreams.Publisher.
List<String> tables = Flowable.fromPublisher(publisher)
    .flatMapIterable(ListTablesResponse::tableNames)
    .toList()
    .blockingGet();
System.out.println(tables);
```

Vedi l'[esempio completo](#) su GitHub.

## Sondaggio sugli stati delle risorse nella versione AWS SDK for Java 2.x: Waiters

L'utilità waiters della versione AWS SDK for Java 2.x consente di verificare che le risorse si trovino in uno stato specifico prima di eseguire operazioni su tali AWS risorse.

Un cameriere è un'astrazione utilizzata per esaminare le AWS risorse, come DynamoDB tabelle o Amazon S3 bucket, fino al raggiungimento dello stato desiderato (o fino a quando non viene stabilito che la risorsa non raggiungerà mai lo stato desiderato). Invece di scrivere una logica per il polling continuo AWS delle risorse, operazione che può essere complessa e soggetta a errori, è possibile utilizzare i camerieri per interrogare una risorsa e far sì che il codice continui a funzionare quando la risorsa è pronta.

## Prerequisiti

[Prima di poter utilizzare i camerieri in un progetto con, è necessario completare i passaggi descritti in Configurazione della versione AWS SDK for Java 2.x. AWS SDK for Java](#)

È inoltre necessario configurare le dipendenze del progetto (ad esempio, nel `build.gradle` file `pom.xml` o) per utilizzare la versione `2.15.0` o successiva di AWS SDK for Java

Per esempio:

```
<project>
  <dependencyManagement>
    <dependencies>
      <dependency>
        <groupId>software.amazon.awssdk</groupId>
        <artifactId>bom</artifactId>
        <version>2.15.0</version>
        <type>pom</type>
        <scope>import</scope>
      </dependency>
    </dependencies>
  </dependencyManagement>
</project>
```

## Usare i camerieri

Per creare un'istanza di un oggetto `waiters`, crea prima un client di servizio. Imposta il `waiter()` metodo del client del servizio come valore dell'oggetto `waiter`. Una volta che l'istanza del cameriere esiste, imposta le sue opzioni di risposta per eseguire il codice appropriato.

## Programmazione sincrona

Il seguente frammento di codice mostra come attendere che una DynamoDB tabella esista e sia in uno stato ATTIVO.

```
DynamoDbClient dynamo = DynamoDbClient.create();
DynamoDbWaiter waiter = dynamo.waiter();

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

// print out the matched response with a tableStatus of ACTIVE
waiterResponse.matched().response().ifPresent(System.out::println);
```

## Programmazione asincrona

Il seguente frammento di codice mostra come attendere che una DynamoDB tabella non esista più.

```

DynamoDbAsyncClient asyncDynamo = DynamoDbAsyncClient.create();
DynamoDbAsyncWaiter asyncWaiter = asyncDynamo.waiter();

CompletableFuture<WaiterResponse<DescribeTableResponse>> waiterResponse =
    asyncWaiter.waitUntilTableNotExists(r -> r.tableName("myTable"));

waiterResponse.whenComplete((r, t) -> {
    if (t == null) {
        // print out the matched ResourceNotFoundException
        r.matched().exception().ifPresent(System.out::println);
    }
}).join();

```

## Configura i camerieri

Puoi personalizzare la configurazione di un cameriere utilizzando il comando `overrideConfiguration()` on its builder. Per alcune operazioni, è possibile applicare una configurazione personalizzata quando si effettua la richiesta.

### Configura un cameriere

Il seguente frammento di codice mostra come sovrascrivere la configurazione su un cameriere.

```

// sync
DynamoDbWaiter waiter =
    DynamoDbWaiter.builder()
        .overrideConfiguration(b -> b.maxAttempts(10))
        .client(dynamoDbClient)
        .build();

// async
DynamoDbAsyncWaiter asyncWaiter =
    DynamoDbAsyncWaiter.builder()
        .client(dynamoDbAsyncClient)
        .overrideConfiguration(o -> o.backoffStrategy(
            FixedDelayBackoffStrategy.create(Duration.ofSeconds(2))))
        .scheduledExecutorService(Executors.newScheduledThreadPool(3))
        .build();

```

### Sostituisci la configurazione per una richiesta specifica

Il seguente frammento di codice mostra come sovrascrivere la configurazione di un cameriere in base alla richiesta. Nota che solo alcune operazioni hanno configurazioni personalizzabili.

```
waiter.waitForTableNotExists(b -> b.tableName("myTable"),
    o -> o.maxAttempts(10));

asyncWaiter.waitForTableExists(b -> b.tableName("myTable"),
    o -> o.waitFor(Duration.ofMinutes(1)));
```

## Esempi di codice

Per un esempio completo di utilizzo di waiters with DynamoDB, consulta [CreateTable.java](#) nel Code Examples Repository. AWS

Per un esempio completo di utilizzo di waiters with Amazon S3, vedete [S3 BucketOps.java](#) nel Code Examples Repository. AWS

## Usa la programmazione asincrona

AWS SDK for Java 2.x Dispone di client asincroni con supporto I/O non bloccante che implementano un'elevata concorrenza su pochi thread. Tuttavia, l'I/O totale non bloccante non è garantito. Il client asincrono può eseguire chiamate di blocco in alcuni casi, ad esempio nel recupero delle credenziali, nella firma delle richieste tramite [AWS Signature Version 4 \(SigV4\)](#) o nel rilevamento degli endpoint.

I metodi sincroni bloccano l'esecuzione del thread finché il client non riceve una risposta dal servizio. I metodi asincroni terminano immediatamente, rassegnando il controllo al thread chiamante senza attendere una risposta.

Poiché un metodo asincrono termina prima che sia disponibile una risposta, occorre un modo per ottenere la risposta quando è pronta. I metodi per il client asincrono nella versione 2.x degli oggetti AWS SDK for Java `CompletableFuture` restituiscono che consentono di accedere alla risposta quando è pronta.

## Operazioni non in streaming

Per le operazioni non di streaming, le chiamate dei metodi asincroni sono simili a quelle dei metodi sincroni. Tuttavia, i metodi asincroni di AWS SDK for Java restituiscono un [CompletableFuture](#) oggetto che contiene i risultati dell'operazione asincrona futura.

Richiamate il `CompletableFuture` `whenComplete()` metodo con un'azione da completare quando il risultato sarà disponibile. `CompletableFuture` implementa l'`Future` interfaccia, quindi puoi ottenere l'oggetto di risposta anche chiamando il `get()` metodo.



Quello che segue è un esempio di operazione asincrona che chiama una Amazon DynamoDB funzione per ottenere un elenco di tabelle, ricevendo una `CompletableFuture` che può contenere un oggetto. [ListTablesResponse](#) L'azione definita nella chiamata a `whenComplete()` viene eseguita solo al termine della chiamata asincrona.

## Importazioni

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbAsyncClient;
import software.amazon.awssdk.services.dynamodb.model.ListTablesRequest;
import software.amazon.awssdk.services.dynamodb.model.ListTablesResponse;
import java.util.List;
import java.util.concurrent.CompletableFuture;
```

## Codice

```
public class DynamoDBAsyncListTables {

    public static void main(String[] args) throws InterruptedException {

        // Create the DynamoDbAsyncClient object
        Region region = Region.US_EAST_1;
        DynamoDbAsyncClient client = DynamoDbAsyncClient.builder()
            .region(region)
            .build();

        listTables(client);
    }

    public static void listTables(DynamoDbAsyncClient client) {

        CompletableFuture<ListTablesResponse> response =
client.listTables(ListTablesRequest.builder()
            .build());

        // Map the response to another CompletableFuture containing just the table
names
        CompletableFuture<List<String>> tableNames =
response.thenApply(ListTablesResponse::tableNames);

        // When future is complete (either successfully or in error) handle the
response
        tableNames.whenComplete((tables, err) -> {
```

```
        try {
            if (tables != null) {
                tables.forEach(System.out::println);
            } else {
                // Handle error
                err.printStackTrace();
            }
        } finally {
            // Lets the application shut down. Only close the client when you are
            // completely done with it.
            client.close();
        }
    });
    tableNames.join();
}
}
```

Nell'esempio di codice riportato di seguito viene illustrato come recuperare una voce da una tabella utilizzando il client asincrono. Invocate il `getItem` metodo di `DynamoDbAsyncClient` e passategli un [GetItemRequest](#) oggetto con il nome della tabella e il valore della chiave primaria dell'elemento desiderato. Questo è in genere il modo in cui si passano i dati richiesti dall'operazione. In questo esempio, viene passato un valore `String`.

## Importazioni

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.model.GetItemRequest;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.DynamoDbAsyncClient;
import java.util.HashMap;
import java.util.Map;
import java.util.Set;
import java.util.stream.Collectors;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
```

## Codice

```
public static void getItem(DynamoDbAsyncClient client, String tableName, String
key, String keyVal) {

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

```
keyToGet.put(key, AttributeValue.builder()
    .s(keyVal).build());

try {

    // Create a GetItemRequest instance
    GetItemRequest request = GetItemRequest.builder()
        .key(keyToGet)
        .tableName(tableName)
        .build();

    // Invoke the DynamoDbAsyncClient object's getItem
    java.util.Collection<AttributeValue> returnedItem =
client.getItem(request).join().item().values();

    // Convert Set to Map
    Map<String, AttributeValue> map =
returnedItem.stream().collect(Collectors.toMap(AttributeValue::s, s->s));
    Set<String> keys = map.keySet();
    for (String sinKey : keys) {
        System.out.format("%s: %s\n", sinKey, map.get(sinKey).toString());
    }

} catch (DynamoDbException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
```

Vedi l'[esempio completo](#) su GitHub

## Operazioni di streaming

Per le operazioni di streaming, è necessario [AsyncRequestBody](#) fornire il contenuto in modo incrementale oppure un modulo [AsyncResponseTransformer](#) per ricevere ed elaborare la risposta.

L'esempio seguente carica un file in modo Amazon S3 asincrono utilizzando l'operazione.

PutObject

Importazioni

```
import software.amazon.awssdk.core.async.AsyncRequestBody;
import software.amazon.awssdk.regions.Region;
```

```
import software.amazon.awssdk.services.s3.S3AsyncClient;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.model.PutObjectResponse;
import java.nio.file.Paths;
import java.util.concurrent.CompletableFuture;
```

## Codice

```
/**
 * To run this AWS code example, ensure that you have setup your development
 * environment, including your AWS credentials.
 *
 * For information, see this documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class S3AsyncOps {

    public static void main(String[] args) {

        final String USAGE = "\n" +
            "Usage:\n" +
            "    S3AsyncOps <bucketName> <key> <path>\n\n" +
            "Where:\n" +
            "    bucketName - the name of the Amazon S3 bucket (for example,
bucket1). \n\n" +
            "    key - the name of the object (for example, book.pdf). \n" +
            "    path - the local path to the file (for example, C:/AWS/book.pdf).
\n" ;

        if (args.length != 3) {
            System.out.println(USAGE);
            System.exit(1);
        }

        String bucketName = args[0];
        String key = args[1];
        String path = args[2];

        Region region = Region.US_WEST_2;
        S3AsyncClient client = S3AsyncClient.builder()
            .region(region)
```

```
        .build();

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

    // Put the object into the bucket
    CompletableFuture<PutObjectResponse> future = client.putObject(objectRequest,
        AsyncRequestBody.fromFile(Paths.get(path))
    );
    future.whenComplete((resp, err) -> {
        try {
            if (resp != null) {
                System.out.println("Object uploaded. Details: " + resp);
            } else {
                // Handle error
                err.printStackTrace();
            }
        } finally {
            // Only close the client when you are completely done with it
            client.close();
        }
    });

    future.join();
}
}
```

L'esempio seguente ottiene un file dalla modalità Amazon S3 asincrona utilizzando l'operazione `GetObject`

### Importazioni

```
import software.amazon.awssdk.core.async.AsyncResponseTransformer;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3AsyncClient;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.GetObjectResponse;
import java.nio.file.Paths;
import java.util.concurrent.CompletableFuture;
```

### Codice

```
/**
 * To run this AWS code example, ensure that you have setup your development
 * environment, including your AWS credentials.
 *
 * For information, see this documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class S3AsyncStreamOps {

    public static void main(String[] args) {

        final String USAGE = "\n" +
            "Usage:\n" +
            "  S3AsyncStreamOps <bucketName> <objectKey> <path>\n\n" +
            "Where:\n" +
            "  bucketName - the name of the Amazon S3 bucket (for example,
bucket1). \n\n" +
            "  objectKey - the name of the object (for example, book.pdf). \n" +
            "  path - the local path to the file (for example, C:/AWS/book.pdf).
\n" ;

        if (args.length != 3) {
            System.out.println(USAGE);
            System.exit(1);
        }

        String bucketName = args[0];
        String objectKey = args[1];
        String path = args[2];

        Region region = Region.US_WEST_2;
        S3AsyncClient client = S3AsyncClient.builder()
            .region(region)
            .build();

        GetObjectRequest objectRequest = GetObjectRequest.builder()
            .bucket(bucketName)
            .key(objectKey)
            .build();
```

```
    CompletableFuture<GetObjectResponse> futureGet =
client.getObject(objectRequest,
    AsyncResponseTransformer.toFile(Paths.get(path)));

    futureGet.whenComplete((resp, err) -> {
        try {
            if (resp != null) {
                System.out.println("Object downloaded. Details: "+resp);
            } else {
                err.printStackTrace();
            }
        } finally {
            // Only close the client when you are completely done with it
            client.close();
        }
    });
    futureGet.join();
}
}
```

## Operazioni avanzate

Il AWS SDK for Java 2.x utilizza [Netty](#), un framework applicativo di rete asincrono basato sugli eventi, per gestire i thread di I/O. Il AWS SDK for Java 2.x crea un `ExecutorService` behind Netty, per completare i futures restituiti dalla richiesta del client HTTP al client Netty. Questa astrazione riduce il rischio che un'applicazione interrompa il processo asincrono se gli sviluppatori scelgono di arrestare o sospendere i thread. Per impostazione predefinita, ogni client asincrono crea un pool di thread in base al numero di processori e gestisce le attività in una coda all'interno di `ExecutorService`.

Gli utenti avanzati possono specificare le dimensioni del pool di thread durante la creazione di un client asincrono utilizzando la seguente opzione di compilazione.

### Codice

```
S3AsyncClient clientThread = S3AsyncClient.builder()
    .asyncConfiguration(
        b -> b.advancedOption(SdkAdvancedAsyncClientOption
            .FUTURE_COMPLETION_EXECUTOR,
            Executors.newFixedThreadPool(10))
    )
    .build();
```

Per ottimizzare le prestazioni, puoi gestire il tuo executor del pool di thread e includerlo durante la configurazione del client.

```
ThreadPoolExecutor executor = new ThreadPoolExecutor(50, 50,
    10, TimeUnit.SECONDS,
    new LinkedBlockingQueue<>(<custom_value>),
    new ThreadFactoryBuilder()
        .threadNamePrefix("sdk-async-response").build());

// Allow idle core threads to time out
executor.allowCoreThreadTimeOut(true);

S3AsyncClient clientThread = S3AsyncClient.builder()
    .asyncConfiguration(
        b -> b.advancedOption(SdkAdvancedAsyncClientOption
            .FUTURE_COMPLETION_EXECUTOR,
            executor
        )
    )
    .build();
```

## Lavora con HTTP/2 in AWS SDK for Java

HTTP/2 è una revisione principale del protocollo HTTP. Questa nuova versione presenta diversi miglioramenti per aumentare le prestazioni:

- La codifica dei dati binari offre un trasferimento dei dati più efficace.
- La compressione delle intestazioni riduce i byte in eccesso scaricati dal client, in modo che ottenga i contenuti in modo più rapido. Questo è particolarmente utile per i client mobili che sono già vincolati dalla larghezza di banda.
- La comunicazione asincrona bidirezionale (multiplexing) consente più richieste e messaggi di risposta tra il client e AWS di essere attivi contemporaneamente su una singola connessione, anziché su più connessioni, il che migliora le prestazioni.

Gli sviluppatori che eseguono l'upgrade agli ultimi SDK utilizzano automaticamente HTTP/2, se il servizio con cui lavorano lo supporta. Le nuove interfacce di programmazione sfruttano in modo ottimale le caratteristiche di HTTP/2 e offrono nuovi modi per creare applicazioni.



La versione AWS SDK for Java 2.x presenta nuove API per lo streaming di eventi che implementano il protocollo HTTP/2. Per esempi di come utilizzare queste nuove API, consulta [Lavorare con Kinesis](#).

## Utilizza le metriche SDK di AWS SDK for Java

Con la AWS SDK for Java versione 2.x, puoi raccogliere metriche sui client di servizio della tua applicazione, analizzare l'output e quindi agire di conseguenza. Amazon CloudWatch

Per impostazione predefinita, la raccolta delle metriche è disabilitata nell'SDK. Questo argomento ti aiuta ad abilitarlo e configurarlo.

### Prerequisiti

Prima di poter abilitare e utilizzare le metriche, devi completare i seguenti passaggi:

- Completa le fasi descritte in [Installazione](#).
- Configura le dipendenze del progetto (ad esempio, nel tuo `build.gradle` file `pom.xml` o) per utilizzare la versione `2.14.0` o successiva di AWS SDK for Java

Per abilitare la pubblicazione delle metriche su CloudWatch, includi anche `cloudwatch-metric-publisherartifactID` con il `2.14.0` numero di versione o successivo nelle dipendenze del progetto.

Per esempio:

```
<project>
  <dependencyManagement>
    <dependencies>
      <dependency>
        <groupId>software.amazon.awssdk</groupId>
        <artifactId>bom</artifactId>
        <version>2.14.0</version>
        <type>pom</type>
        <scope>import</scope>
      </dependency>
    </dependencies>
  </dependencyManagement>
  <dependencies>
    <dependency>
      <groupId>software.amazon.awssdk</groupId>
```

```
<artifactId>cloudwatch-metric-publisher</artifactId>
<version>2.14.0</version>
</dependency>
</dependencies>
</project>
```

- Abilita `cloudwatch:PutMetricData` le autorizzazioni per l'identità IAM utilizzata dall'editore delle metriche per consentire all'SDK for Java di scrivere metriche.

## Come abilitare la raccolta delle metriche

Puoi abilitare le metriche nella tua applicazione per un client di servizio o su richieste individuali.

### Abilita le metriche per una richiesta specifica

La classe seguente mostra come abilitare l'editore delle CloudWatch metriche per una richiesta a Amazon DynamoDB Utilizza la configurazione predefinita di Metrics Publisher.

```
import software.amazon.awssdk.metrics.MetricPublisher;
import software.amazon.awssdk.metrics.publishers.cloudwatch.CloudWatchMetricPublisher;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.ListTablesRequest;

public class DefaultConfigForRequest {
    // Use one MetricPublisher for your application. It can be used with requests or
    // service clients.
    static MetricPublisher metricsPub = CloudWatchMetricPublisher.create();

    public static void main(String[] args) {
        DynamoDbClient ddb = DynamoDbClient.create();
        // Publish metrics the for ListTables operation.
        ddb.listTables(ListTablesRequest.builder()
            .overrideConfiguration(c -> c.addMetricPublisher(metricsPub))
            .build());

        // Perform more work in your application.

        // A MetricsPublisher has its own lifecycle independent of any service client
        // or request that uses it.
        // If you no longer need the publisher, close it to free up resources.
        metricsPub.close(); // All metrics stored in memory are flushed to CloudWatch.
    }
}
```

```
        // Perform more work with the DynamoDbClient instance without publishing
metrics.
        // Close the service client when you no longer need it.
        ddb.close();
    }
}
```

### Important

Assicurati che l'applicazione richiami `close` l'[MetricPublisher](#) istanza quando il client del servizio non è più in uso. In caso contrario, potrebbero verificarsi fughe di thread o descrittori di file.

## Abilita le metriche di riepilogo per un client di servizio specifico

Il seguente frammento di codice mostra come abilitare un editore di CloudWatch metriche con impostazioni predefinite per un client di servizio.

```
MetricPublisher metricsPub = CloudWatchMetricPublisher.create();

DynamoDbClient ddb = DynamoDbClient.builder()
    .overrideConfiguration(c -> c.addMetricPublisher(metricsPub))
    .build();
```

## Personalizza l'editore di metriche

La seguente classe dimostra come impostare una configurazione personalizzata per l'editore delle metriche per uno specifico client di servizio. Le personalizzazioni includono il caricamento di un profilo specifico, la specificazione di una AWS regione in cui l'editore delle metriche invia le richieste e la personalizzazione della frequenza a cui l'editore invia le metriche. CloudWatch

```
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.core.metrics.CoreMetric;
import software.amazon.awssdk.metrics.MetricPublisher;
import software.amazon.awssdk.metrics.publishers.cloudwatch.CloudWatchMetricPublisher;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudwatch.CloudWatchAsyncClient;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;

import java.time.Duration;
```

```
public class CustomConfigForDDBClient {
    // Use one MetricPublisher for your application. It can be used with requests or
    // service clients.
    static MetricPublisher metricsPub = CloudWatchMetricPublisher.builder()
        .cloudWatchClient(CloudWatchAsyncClient.builder()
            .region(Region.US_WEST_2)
            .credentialsProvider(ProfileCredentialsProvider.create("cloudwatch"))
            .build())
        .uploadFrequency(Duration.ofMinutes(5))
        .maximumCallsPerUpload(100)
        .namespace("ExampleSDKV2Metrics")
        .detailedMetrics(CoreMetric.API_CALL_DURATION)
        .build();

    public static void main(String[] args) {
        DynamoDbClient ddb = DynamoDbClient.builder()
            .overrideConfiguration(c -> c.addMetricPublisher(metricsPub))
            .build();
        // Publish metrics for DynamoDB operations.
        ddb.listTables();
        ddb.describeEndpoints();
        ddb.describeLimits();
        // Perform more work in your application.

        // A MetricsPublisher has its own lifecycle independent of any service client
        // or request that uses it.
        // If you no longer need the publisher, close it to free up resources.
        metricsPub.close(); // All metrics stored in memory are flushed to CloudWatch.

        // Perform more work with the DynamoDbClient instance without publishing
        // metrics.
        // Close the service client when you no longer need it.
        ddb.close();
    }
}
```

Le personalizzazioni mostrate nello snippet precedente hanno i seguenti effetti.

- Il `cloudWatchClient` metodo consente di personalizzare il CloudWatch client utilizzato per inviare le metriche. In questo esempio, utilizziamo una regione diversa da quella predefinita di `us-east-1` in cui il client invia le metriche. Utilizziamo anche un profilo con nome diverso, `cloudwatch`,

le cui credenziali verranno utilizzate per autenticare le richieste. CloudWatch Tali credenziali devono avere le autorizzazioni per `cloudwatch:PutMetricData`

- Il `uploadFrequency` metodo consente di specificare la frequenza con cui l'editore di metriche carica le metriche. CloudWatch L'impostazione predefinita è una volta al minuto.
- Il `maximumCallsPerUpload` metodo limita il numero di chiamate effettuate per ogni caricamento. Il valore predefinito è illimitato.
- Per impostazione predefinita, l'SDK for Java 2.x pubblica le metriche nello spazio dei nomi. `AwsSdk/JavaSdk2` È possibile utilizzare il `namespace` metodo per specificare un valore diverso.
- Per impostazione predefinita, l'SDK pubblica metriche di riepilogo. Le metriche di riepilogo sono costituite da `media`, `minima`, `massima`, `somma` e `numero di campioni`. Specificando una o più metriche SDK nel `detailedMetrics` metodo, l'SDK pubblica dati aggiuntivi per ogni metrica. Questi dati aggiuntivi consentono statistiche percentili come `p90` e `p99` su cui è possibile eseguire interrogazioni. CloudWatch Le metriche dettagliate sono particolarmente utili per le metriche di latenza come `APICallDuration`, che misurano la latenza per le richieste dei end-to-end client SDK. Puoi utilizzare i campi della [CoreMetric](#) classe per specificare altre metriche SDK comuni.

## Quando sono disponibili le metriche?

Le metriche sono generalmente disponibili entro 5-10 minuti dall'emissione da parte dell'SDK for Java. Per informazioni accurate e up-to-date metriche, controlla Cloudwatch almeno 10 minuti dopo aver emesso le metriche delle tue applicazioni Java.

## Quali informazioni vengono raccolte?

La raccolta di metriche include quanto segue:

- Numero di richieste API, incluso se hanno esito positivo o negativo
- Informazioni sui AWS servizi richiamati nelle richieste API, incluse le eccezioni restituite
- La durata di varie operazioni come `Marshalling`, `Signing` e richieste HTTP
- Metriche del client HTTP, come il numero di connessioni aperte, il numero di richieste in sospeso e il nome del client HTTP utilizzato

### Note

Le metriche disponibili variano in base al client HTTP.

Per un elenco completo, consulta le [metriche del client di servizio](#).

## Come posso usare queste informazioni?

Puoi utilizzare le metriche raccolte dall'SDK per monitorare i client di servizio nella tua applicazione. Puoi esaminare le tendenze generali di utilizzo, identificare anomalie, esaminare le eccezioni restituite dai client di servizio o approfondire per comprendere un problema particolare. Utilizzando Amazon CloudWatch, puoi anche creare allarmi per avvisarti non appena l'applicazione raggiunge una condizione da te definita.

[Per ulteriori informazioni, consulta Utilizzo Amazon CloudWatch delle metriche e Utilizzo degli Amazon CloudWatch allarmi nella Guida per l'Amazon CloudWatch utente.](#)

## Metriche dei client di assistenza

[Con AWS SDK for Java 2.x, puoi raccogliere metriche dai client di servizio nella tua applicazione e quindi pubblicarle \(emettere\) tali metriche su Amazon. CloudWatch](#)

Queste tabelle elencano le metriche che puoi raccogliere e qualsiasi requisito di utilizzo del client HTTP.

[Per ulteriori informazioni sull'abilitazione e la configurazione delle metriche per l'SDK, consulta Abilitazione delle metriche SDK.](#)

## Metriche raccolte con ogni richiesta

Nome parametro	Descrizione	Type
ApiCallDuration	Il tempo totale impiegato per completare una richiesta (inclusi tutti i nuovi tentativi).	Durata
ApiCallSuccessful	Vero se la chiamata API ha avuto esito positivo; falso in caso contrario.	Booleano
CredentialsFetchDuration	Il tempo impiegato per recuperare le credenziali di AWS firma per la richiesta.	Durata

Nome parametro	Descrizione	Type
EndpointResolveDuration	Il tempo impiegato per risolvere l'endpoint utilizzato per la chiamata API.	Durata
MarshallingDuration	Il tempo necessario per adattare una richiesta SDK a una richiesta HTTP.	Durata
OperationName	Il nome dell' AWS API a cui viene effettuata la richiesta.	Stringa
RetryCount	Numero di volte in cui l'SDK ha ripetuto la chiamata API.	Numero intero
ServiceId	ID del servizio su Servizio AWS cui viene effettuata la richiesta API.	Stringa
TokenFetchDuration	Il tempo impiegato per recuperare le credenziali di firma dei token per la richiesta.	Durata

## Metriche raccolte per ogni tentativo di richiesta

Ogni chiamata API potrebbe richiedere più tentativi prima di ricevere una risposta. Queste metriche vengono raccolte per ogni tentativo.

### Metriche principali

Nome parametro	Descrizione	Type
AwsExtendedRequestId	L'ID della richiesta estesa della richiesta di servizio.	Stringa
AwsRequestId	L'ID della richiesta di servizio.	Stringa

Nome parametro	Descrizione	Type
BackoffDelayDuration	Il periodo di attesa dell'SDK prima di questo tentativo di chiamata API.	Durata
ErrorType	Il tipo di errore che si è verificato durante un tentativo di chiamata.	Stringa
ReadThroughput	La velocità di lettura del client.	Doppio
ServiceCallDuration	Il tempo necessario per connettersi al servizio, inviare la richiesta e ricevere il codice di stato HTTP e l'intestazione dalla risposta.	Durata
SigningDuration	Il tempo necessario per firmare la richiesta HTTP.	Durata
TimeToFirstByte	Tempo trascorso dall'invio della richiesta HTTP (inclusa l'acquisizione di una connessione) alla ricezione del primo byte delle intestazioni della risposta.	Durata
TimeToLastByte	Tempo trascorso dall'invio della richiesta HTTP (inclusa l'acquisizione di una connessione) alla ricezione dell'ultimo byte della risposta.	Durata
UnmarshallingDuration	Il tempo necessario per annullare il marshall di una risposta HTTP a una risposta SDK.	Durata



## Metriche HTTP

Nome parametro	Descrizione	Type	Client HTTP richiesto*
AvailableConcurrency	Il numero di richieste concorrenti rimanenti che possono essere supportate dal client HTTP senza dover stabilire un'altra connessione.	Numero intero	Apache, Netty, CRT
ConcurrencyAcquireDuration	Il tempo impiegato per acquisire un canale dal pool di connessioni.	Durata	Apache, Netty, CRT
HttpClientName	Il nome dell'HTTP utilizzato per la richiesta.	Stringa	Apache, Netty, CRT
HttpStatusCode	Il codice di stato restituito con la risposta HTTP.	Numero intero	Qualsiasi
LeasedConcurrency	Il numero di richieste attualmente eseguite dal client HTTP.	Numero intero	Apache, Netty, CRT
LocalStreamWindowSize	La dimensione della finestra HTTP/2 locale in byte per il flusso su cui è stata eseguita questa richiesta.	Numero intero	Netty
MaxConcurrency	Il numero massimo di richieste simultanee	Numero intero	Apache, Netty, CRT

Nome parametro	Descrizione	Type	Client HTTP richiesto*
	supportate dal client HTTP.		
PendingConcurrency Acquires	Il numero di richieste bloccate, in attesa che un'altra connessione TCP o un nuovo stream sia disponibile dal pool di connessioni.	Numero intero	Apache, Netty, CRT
RemoteStreamWindowSize	La dimensione della finestra HTTP/2 remota in byte per il flusso su cui è stata eseguita questa richiesta.	Numero intero	Netty

I termini usati nella colonna significano:

- Apache: il client HTTP basato su Apache () [ApacheHttpClient](#)
- Netty: il client HTTP basato su Netty () [NettyNioAsyncHttpClient](#)
- CRT: il client HTTP basato su AWS CRT () [AwsCrtAsyncHttpClient](#)
- Qualsiasi: la raccolta di dati metrici non dipende dal client HTTP, incluso il client HTTP basato su URLConnection () [URLConnectionHttpClient](#)

# Lavora con Servizi AWS l'utilizzo di AWS SDK for Java 2.x

Questa sezione fornisce brevi tutorial e indicazioni su come lavorare con select. Servizi AWS Per un set completo di esempi, consulta la sezione [Esempi di codice](#).

## Argomenti

- [Utilizzo delle CloudWatch](#)
- [AWS servizi di database e AWS SDK for Java 2.x](#)
- [Lavora con DynamoDB](#)
- [Lavora con Amazon EC2](#)
- [Utilizzo delle IAM](#)
- [Lavora con Kinesis](#)
- [Funzioni di richiamo, elenco ed eliminazione AWS Lambda](#)
- [Funziona con Amazon S3](#)
- [Utilizzo delle Amazon Simple Notification Service](#)
- [Lavora con Amazon Simple Queue Service](#)
- [Utilizzo delle Amazon Transcribe](#)

## Utilizzo delle CloudWatch

Questa sezione fornisce esempi di programmazione di [Amazon CloudWatch](#) utilizzando AWS SDK for Java 2.x.

Amazon CloudWatch monitora in tempo reale le risorse Amazon Web Services (AWS) e le applicazioni in esecuzione su AWS. Puoi utilizzare CloudWatch per raccogliere e monitorare parametri, che sono variabili che puoi misurare per le risorse e le applicazioni. Gli allarmi CloudWatch inviano notifiche o apportano automaticamente modifiche alle risorse che stai monitorando in base a regole da te definite.

Gli esempi seguenti includono solo il codice necessario per dimostrare ciascuna tecnica. Il [codice di esempio completo è disponibile su GitHub](#) Da qui puoi scaricare un singolo file sorgente o clonare l'archivio localmente per ottenere tutti gli esempi da creare ed eseguire.

## Argomenti

- [Ottieni metriche da CloudWatch](#)
- [Pubblica dati metrici personalizzati su CloudWatch](#)
- [Lavora con gli CloudWatch allarmi](#)
- [Usa Amazon CloudWatch Events](#)

## Ottieni metriche da CloudWatch

### Elencazione dei parametri

Per elencare le CloudWatch metriche, crea un metodo [ListMetricsRequeste](#) chiama il CloudWatchClient metodo's. listMetrics Puoi utilizzare ListMetricsRequest per filtrare i parametri restituiti in base a spazio dei nomi, nome parametro o dimensioni.

#### Note

Un elenco di metriche e dimensioni pubblicate dai AWS servizi è disponibile nel [riferimento alle Amazon CloudWatch metriche e alle dimensioni](#) nella Guida per l'Amazon CloudWatchutente.

### Importazioni

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudwatch.CloudWatchClient;
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatch.model.ListMetricsRequest;
import software.amazon.awssdk.services.cloudwatch.model.ListMetricsResponse;
import software.amazon.awssdk.services.cloudwatch.model.Metric;
```

### Codice

```
public static void listMets( CloudWatchClient cw, String namespace) {

    boolean done = false;
    String nextToken = null;

    try {
        while(!done) {
```

```
ListMetricsResponse response;

if (nextToken == null) {
    ListMetricsRequest request = ListMetricsRequest.builder()
        .namespace(namespace)
        .build();

    response = cw.listMetrics(request);
} else {
    ListMetricsRequest request = ListMetricsRequest.builder()
        .namespace(namespace)
        .nextToken(nextToken)
        .build();

    response = cw.listMetrics(request);
}

for (Metric metric : response.metrics()) {
    System.out.printf(
        "Retrieved metric %s", metric.metricName());
    System.out.println();
}

if(response.nextToken() == null) {
    done = true;
} else {
    nextToken = response.nextToken();
}
}

} catch (CloudWatchException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

Le metriche vengono restituite in un [ListMetricsResponse](#) chiamando il relativo metodo `getMetrics`

I risultati possono essere paginati. Per recuperare il batch di risultati successivo, chiama `nextToken` sull'oggetto di risposta e utilizza il valore del token per creare un nuovo oggetto richiesta. Quindi, chiama nuovamente il metodo `listMetrics` con una nuova richiesta.

Vedi l'[esempio completo](#) su GitHub

## Ulteriori informazioni

- [ListMetrics](#) nell'Amazon CloudWatch API Reference

## Pubblica dati metrici personalizzati su CloudWatch

Alcuni AWS servizi pubblicano [le proprie metriche](#) in namespace che iniziano con "AWS". Puoi anche pubblicare dati metrici personalizzati utilizzando il tuo spazio dei nomi (purché non inizi con "«"). AWS

### Pubblicare dati dei parametri personalizzati

Per pubblicare i tuoi dati metrici, chiama il metodo's con a. `CloudWatchClient.putMetricData` [PutMetricDataRequest](#) `PutMetricDataRequest` Devono includere lo spazio dei nomi personalizzato da utilizzare per i dati e le informazioni sul punto dati stesso in un oggetto.

#### [MetricDatum](#)

#### Note

Non è possibile specificare uno spazio dei nomi che inizia con "»". AWS I namespace che iniziano con "AWS" sono riservati all'uso da parte dei prodotti. Amazon Web Services

## Importazioni

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudwatch.CloudWatchClient;
import software.amazon.awssdk.services.cloudwatch.model.Dimension;
import software.amazon.awssdk.services.cloudwatch.model.MetricDatum;
import software.amazon.awssdk.services.cloudwatch.model.StandardUnit;
import software.amazon.awssdk.services.cloudwatch.model.PutMetricDataRequest;
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import java.time.Instant;
import java.time.ZoneOffset;
import java.time.ZonedDateTime;
import java.time.format.DateTimeFormatter;
```

## Codice

```
public static void putMetData(CloudWatchClient cw, Double dataPoint ) {
```

```
try {
    Dimension dimension = Dimension.builder()
        .name("UNIQUE_PAGES")
        .value("URLS")
        .build();

    // Set an Instant object
    String time =
ZonedDateTime.now( ZoneOffset.UTC ).format( DateTimeFormatter.ISO_INSTANT );
    Instant instant = Instant.parse(time);

    MetricDatum datum = MetricDatum.builder()
        .metricName("PAGES_VISITED")
        .unit(StandardUnit.NONE)
        .value(dataPoint)
        .timestamp(instant)
        .dimensions(dimension).build();

    PutMetricDataRequest request = PutMetricDataRequest.builder()
        .namespace("SITE/TRAFFIC")
        .metricData(datum).build();

    cw.putMetricData(request);

} catch (CloudWatchException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
System.out.printf("Successfully put data point %f", dataPoint);
}
```

Vedi l'esempio [completo](#) su GitHub

## Ulteriori informazioni

- [Usa le Amazon CloudWatch metriche](#) nella Guida per l'Amazon CloudWatchutente.
- [AWSNamespace nella Guida per l'utente](#). Amazon CloudWatch
- [PutMetricData](#) nell'API Reference. Amazon CloudWatch

# Lavora con gli CloudWatch allarmi

## Creazione di un allarme

Per creare un allarme basato su una CloudWatch metrica, chiama il `putMetricAlarm` metodo `CloudWatchClient`'s con una [PutMetricAlarmRequest](#) compilazione delle condizioni di allarme.

### Importazioni

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudwatch.CloudWatchClient;
import software.amazon.awssdk.services.cloudwatch.model.Dimension;
import software.amazon.awssdk.services.cloudwatch.model.PutMetricAlarmRequest;
import software.amazon.awssdk.services.cloudwatch.model.ComparisonOperator;
import software.amazon.awssdk.services.cloudwatch.model.Statistic;
import software.amazon.awssdk.services.cloudwatch.model.StandardUnit;
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
```

### Codice

```
public static void putMetricAlarm(CloudWatchClient cw, String alarmName, String
instanceId) {

    try {
        Dimension dimension = Dimension.builder()
            .name("InstanceId")
            .value(instanceId).build();

        PutMetricAlarmRequest request = PutMetricAlarmRequest.builder()
            .alarmName(alarmName)
            .comparisonOperator(
                ComparisonOperator.GREATER_THAN_THRESHOLD)
            .evaluationPeriods(1)
            .metricName("CPUUtilization")
            .namespace("AWS/EC2")
            .period(60)
            .statistic(Statistic.AVERAGE)
            .threshold(70.0)
            .actionsEnabled(false)
            .alarmDescription(
                "Alarm when server CPU utilization exceeds 70%")
            .unit(StandardUnit.SECONDS)
            .dimensions(dimension)
```



```
        .build();

        cw.putMetricAlarm(request);
        System.out.printf(
            "Successfully created alarm with name %s", alarmName);
    } catch (CloudWatchException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

Guarda l'[esempio completo](#) su GitHub

## Elencare allarmi

Per elencare gli CloudWatch allarmi che hai creato, chiama il `describeAlarms` metodo `CloudWatchClient`'s con un [DescribeAlarmsRequest](#) che puoi usare per impostare le opzioni per il risultato.

### Importazioni

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudwatch.CloudWatchClient;
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatch.model.DescribeAlarmsRequest;
import software.amazon.awssdk.services.cloudwatch.model.DescribeAlarmsResponse;
import software.amazon.awssdk.services.cloudwatch.model.MetricAlarm;
```

### Codice

```
public static void desCWAAlarms( CloudWatchClient cw) {

    try {

        boolean done = false;
        String newToken = null;

        while(!done) {
            DescribeAlarmsResponse response;

            if (newToken == null) {
```

```
        DescribeAlarmsRequest request =
DescribeAlarmsRequest.builder().build();
        response = cw.describeAlarms(request);
    } else {
        DescribeAlarmsRequest request = DescribeAlarmsRequest.builder()
            .nextToken(newToken)
            .build();
        response = cw.describeAlarms(request);
    }

    for(MetricAlarm alarm : response.metricAlarms()) {
        System.out.printf("\n Retrieved alarm %s", alarm.alarmName());
    }

    if(response.nextToken() == null) {
        done = true;
    } else {
        newToken = response.nextToken();
    }
}

} catch (CloudWatchException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
System.out.printf("Done");
}
```

L'elenco degli allarmi può essere ottenuto `MetricAlarms` chiamando il comando [DescribeAlarmsResponse](#) che viene restituito da `describeAlarms`

I risultati possono essere paginati. Per recuperare il batch di risultati successivo, chiama `nextToken` sull'oggetto di risposta e utilizza il valore del token per creare un nuovo oggetto richiesta. Quindi, chiama nuovamente il metodo `describeAlarms` con una nuova richiesta.

#### Note

Puoi anche recuperare gli allarmi per una metrica specifica utilizzando il metodo `CloudWatchClient describeAlarmsForMetric`. L'uso è simile a `describeAlarms`.

Vedi l'esempio [completo](#) su GitHub

## Eliminare allarmi

Per eliminare gli CloudWatch allarmi, chiama il `deleteAlarms` metodo `CloudWatchClient`'s [DeleteAlarmsRequest](#) contenente uno o più nomi di allarmi che desideri eliminare.

### Importazioni

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudwatch.CloudWatchClient;
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatch.model.DeleteAlarmsRequest;
```

### Codice

```
public static void deleteCWAlarm(CloudWatchClient cw, String alarmName) {

    try {
        DeleteAlarmsRequest request = DeleteAlarmsRequest.builder()
            .alarmNames(alarmName)
            .build();

        cw.deleteAlarms(request);
        System.out.printf("Successfully deleted alarm %s", alarmName);

    } catch (CloudWatchException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

Guarda l'[esempio completo](#) su. GitHub

### Ulteriori informazioni

- [Usò degli Amazon CloudWatch allarmi](#) nella Guida per l'Amazon CloudWatchutente
- [PutMetricAlarm](#) nell'Amazon CloudWatchAPI Reference
- [DescribeAlarms](#) nell'Amazon CloudWatchAPI Reference
- [DeleteAlarms](#) nell'Amazon CloudWatchAPI Reference

## Usa Amazon CloudWatch Events

CloudWatch Events fornisce un flusso quasi in tempo reale di eventi di sistema che descrivono le modifiche nelle AWS risorse relative a Amazon EC2 istanze, Lambda funzioni, Kinesis flussi, Amazon ECS attività, macchine a Step Functions stati, Amazon SNS argomenti, Amazon SQS code o destinazioni integrate. Puoi abbinare gli eventi e instradarli verso una o più funzioni o stream target utilizzando regole semplici.

Amazon EventBridge è l'[evoluzione](#) di CloudWatch Events. Entrambi i servizi utilizzano la stessa API, quindi puoi continuare a utilizzare il [client CloudWatch Events](#) fornito dall'SDK o migrare all'SDK per la funzionalità [EventBridge client](#) for CloudWatch Events di Java. CloudWatch [La documentazione della Events User Guide](#) e i [riferimenti all'API](#) sono ora disponibili nei siti di documentazione.

EventBridge

### Aggiunta di eventi

Per aggiungere CloudWatch eventi personalizzati, chiamate il `CloudWatchEventsClient`'s `putEvents` metodo con un [PutEventsRequest](#) oggetto che contiene uno o più [PutEventsRequestEntry](#) oggetti che forniscono dettagli su ogni evento. Puoi specificare diversi parametri per la voce, ad esempio l'origine e il tipo di evento, le risorse associate all'evento e così via.

#### Note

Puoi specificare un massimo di 10 eventi per chiamata a `putEvents`.

### Importazioni

```
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatchevents.CloudWatchEventsClient;
import software.amazon.awssdk.services.cloudwatchevents.model.PutEventsRequest;
import software.amazon.awssdk.services.cloudwatchevents.model.PutEventsRequestEntry;
```

### Codice

```
public static void putCWEvents(CloudWatchEventsClient cwe, String resourceArn ) {

    try {

        final String EVENT_DETAILS =
```

```
        "{ \"key1\": \"value1\", \"key2\": \"value2\" }";

        PutEventsRequestEntry requestEntry = PutEventsRequestEntry.builder()
            .detail(EVENT_DETAILS)
            .detailType("sampleSubmitted")
            .resources(resourceArn)
            .source("aws-sdk-java-cloudwatch-example")
            .build();

        PutEventsRequest request = PutEventsRequest.builder()
            .entries(requestEntry)
            .build();

        cwe.putEvents(request);
        System.out.println("Successfully put CloudWatch event");

    } catch (CloudWatchException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

Vedi l'[esempio completo](#) su GitHub.

## Aggiunta di regole

Per creare o aggiornare una regola, chiamate il `CloudWatchEventsClient`'s `putRule` metodo con a [PutRuleRequest](#) con il nome della regola e parametri opzionali come il [modello di evento](#), il IAM ruolo da associare alla regola e un'[espressione di pianificazione](#) che descriva la frequenza con cui la regola viene eseguita.

### Importazioni

```
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatchevents.CloudWatchEventsClient;
import software.amazon.awssdk.services.cloudwatchevents.model.PutRuleRequest;
import software.amazon.awssdk.services.cloudwatchevents.model.PutRuleResponse;
import software.amazon.awssdk.services.cloudwatchevents.model.RuleState;
```

### Codice

```
public static void putCWRule(CloudWatchEventsClient cwe, String ruleName, String
roleArn) {
```

```
try {
    PutRuleRequest request = PutRuleRequest.builder()
        .name(ruleName)
        .roleArn(roleArn)
        .scheduleExpression("rate(5 minutes)")
        .state(RuleState.ENABLED)
        .build();

    PutRuleResponse response = cwe.putRule(request);
    System.out.printf(
        "Successfully created CloudWatch events rule %s with arn %s",
        roleArn, response.ruleArn());
} catch (
    CloudWatchException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

Vedi l'[esempio completo](#) su GitHub

## Aggiunta di target

I target sono le risorse che vengono invocate quando una regola viene attivata. Gli obiettivi di esempio includono Amazon EC2 istanze, Lambda funzioni, Kinesis flussi, Amazon ECS attività, macchine a Step Functions stati e destinazioni integrate.

Per aggiungere un obiettivo a una regola, chiama il `CloudWatchEventsClient`'s `putTargets` metodo [PutTargetsRequest](#) contenente la regola da aggiornare e un elenco di obiettivi da aggiungere alla regola.

## Importazioni

```
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatchevents.CloudWatchEventsClient;
import software.amazon.awssdk.services.cloudwatchevents.model.PutTargetsRequest;
import software.amazon.awssdk.services.cloudwatchevents.model.PutTargetsResponse;
import software.amazon.awssdk.services.cloudwatchevents.model.Target;
```

## Codice

```
public static void putCWTargets(CloudWatchEventsClient cwe, String ruleName, String
functionArn, String targetId ) {

    try {
        Target target = Target.builder()
            .arn(functionArn)
            .id(targetId)
            .build();

        PutTargetsRequest request = PutTargetsRequest.builder()
            .targets(target)
            .rule(ruleName)
            .build();

        PutTargetsResponse response = cwe.putTargets(request);
        System.out.printf(
            "Successfully created CloudWatch events target for rule %s",
            ruleName);
    } catch (CloudWatchException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

Vedi l'[esempio completo](#) su GitHub.

## Ulteriori informazioni

- [Aggiungere eventi con PutEvents](#) la Amazon EventBridge User Guide
- [Pianifica le espressioni per le regole](#) nella Amazon EventBridge User Guide
- [Tipi di eventi indicati CloudWatch Events](#) nella Amazon EventBridge User Guide
- [Modelli di eventi](#) nella Amazon EventBridge User Guide
- [PutEvents](#) nell'Amazon EventBridge API Reference
- [PutTargets](#) nell'Amazon EventBridge API Reference
- [PutRule](#) nell'Amazon EventBridge API Reference

## AWS servizi di database e AWS SDK for Java 2.x

AWS [offre diversi tipi di database: relazionali, chiave-valore, in memoria, documentali e molti altri](#). Il supporto dell'SDK for Java 2.x varia a seconda della natura del servizio di database in uso. AWS

Alcuni servizi di database, ad esempio il servizio [Amazon](#) DynamoDB, dispongono di API di servizi Web per gestire AWS la risorsa (database) e di API di servizi Web per interagire con i dati. [Nell'SDK for Java 2.x questi tipi di servizi dispongono di client di servizio dedicati, ad esempio DynamoDBClient.](#)

Altri servizi di database dispongono di API di servizi Web che interagiscono con la risorsa, come l'API [Amazon](#) DocumentDB (per la gestione di cluster, istanze e risorse), ma non dispongono di un'API di servizio Web per l'utilizzo dei dati. L'SDK for Java 2.x ha un'interfaccia [DocDbClient](#) corrispondente per lavorare con la risorsa. Tuttavia, è necessaria un'altra API Java, come [MongoDB per Java, per](#) lavorare con i dati.

Utilizza gli esempi seguenti per scoprire come utilizzare i client di servizio SDK for Java 2.x con i diversi tipi di database.

### Esempi di Amazon DynamoDB

Lavorare con i dati

Client di servizio SDK: [DynamoDbClient](#)

Esempio: applicazione [REST React/Spring](#) che utilizza DynamoDB

Esempi: [diversi esempi di DynamoDB](#)

Client di servizio SDK: [DynamoDbEnhancedClient](#)

Esempio: applicazione [REST React/Spring](#) che utilizza DynamoDB

Esempi: [diversi esempi di DynamoDB](#) (nomi che iniziano con «Enhanced»)

Lavorare con il database

Client di servizio SDK: [DynamoDbClient](#)

[CreateTableEsempi ListTables:](#), DeleteTable



Vedi [altri esempi di DynamoDB](#) nella sezione degli esempi di codice guidati di questa guida.

## Esempi di Amazon RDS

Lavorare con i dati	Lavorare con il database
API non SDK: JDBC, versione SQL specifica del database; il codice gestisce le connessioni al database o un pool di connessioni.	Client di servizio SDK: <a href="#">RdsClient</a>
Esempio: applicazione <a href="#">REST React/Spring</a> che utilizza MySQL	<a href="#">Esempi RdsClient</a> : diversi esempi

## Esempi di Amazon Redshift

Utilizzo dei dati	Lavorare con il database
Client di servizio SDK: <a href="#">RedshiftDataClient</a>	Client di servizio SDK: <a href="#">RedshiftClient</a>
Esempi: <a href="#">diversi RedshiftDataClient</a> esempi	Esempi: <a href="#">diversi RedshiftClient</a> esempi
Esempio: applicazione <a href="#">REST React/Spring</a> che utilizza RedshiftDataClient	

## Esempi di Amazon Aurora Serverless v2

Lavorare con i dati	Lavorare con il database
Client di servizio SDK: <a href="#">RdsDataClient</a>	Client di servizio SDK: <a href="#">RdsClient</a>
Esempio: applicazione <a href="#">REST React/Spring</a> che utilizza RdsDataClient	<a href="#">Esempi: diversi esempi RdsClient</a>

## Esempi di Amazon DocumentDB

Lavorare con i dati	Lavorare con il database
API non SDK: libreria Java specifica per MongoDB (ad esempio MongoDB per <a href="#">Java</a> ); <a href="#">il codice gestisce le connessioni</a> al database o un pool di connessioni.	Client di servizio SDK: <a href="#">DocDbClient</a>
Esempi: <a href="#">DocumentDB (Mongo) Developer Guide</a> (seleziona la scheda 'Java')	

## Lavora con DynamoDB

Questa sezione fornisce esempi che mostrano come lavorare con [DynamoDB](#).

I seguenti esempi utilizzano il client DynamoDB standard di basso livello [DynamoDbClient](#) () del 2.x. AWS SDK for Java

- [the section called “Lavora con le tabelle in DynamoDB”](#)
- [the section called “Lavora con gli elementi in DynamoDB”](#)

L'SDK offre anche il [DynamoDB Enhanced](#) Client che offre un approccio di alto livello orientato agli oggetti per lavorare con DynamoDB. La sezione seguente illustra questo client in modo approfondito.

- [the section called “ Mappa gli oggetti sugli elementi di DynamoDB”](#)

## Lavora con le tabelle in DynamoDB

Le tabelle sono i contenitori per tutti gli elementi di un DynamoDB database. Prima di poter aggiungere o rimuovere dati da DynamoDB, è necessario creare una tabella.

Per ogni tabella, devi definire:

- Un nome di tabella unico per il tuo account e la tua regione.
- Una chiave primaria per la quale ogni valore deve essere univoco; due item nella tabella non possono avere lo stesso valore della chiave primaria.

La chiave primaria può essere semplice, costituita da una singola chiave di partizione (HASH), o composta, costituita da una chiave di partizione e una di ordinamento (RANGE).

A ogni valore chiave è associato un tipo di dati, enumerato dalla classe. [ScalarAttributeType](#)  
Il valore della chiave può essere binario (B), numerico (N) o una stringa (S). Per ulteriori informazioni, consulta [Regole di denominazione e tipi di dati](#) nella Guida per gli sviluppatori.

Amazon DynamoDB

- Il throughput assegnato è costituito da valori che definiscono il numero di unità di capacità di lettura/scrittura prenotati per la tabella.

#### Note

[Amazon DynamoDB i prezzi](#) si basano sui valori di throughput assegnati che imposti sulle tabelle, quindi prenota solo la capacità che ritieni necessaria per la tabella.

Il throughput assegnato per una tabella può essere modificato in qualsiasi momento, pertanto puoi modificare la capacità in base alle esigenze.

## Creare una tabella

Usa il `DynamoDbClient`'s `createTable` metodo per creare una nuova DynamoDB tabella. È necessario costruire gli attributi della tabella e uno schema della tabella, entrambi utilizzati per identificare la chiave primaria della tabella. Inoltre, occorre fornire i valori del throughput assegnato iniziali e un nome della tabella.

#### Note

Se esiste già una tabella con il nome scelto, ne [DynamoDbException](#) viene generata una.

## Creazione di una tabella con una chiave primaria semplice

Questo codice crea una tabella con un attributo che è la chiave primaria semplice della tabella. l'esempio utilizza [AttributeDefinition](#) e [KeySchemaElement](#) oggetti per. [CreateTableRequest](#)

## Importazioni

```
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.model.CreateTableRequest;
import software.amazon.awssdk.services.dynamodb.model.AttributeDefinition;
import software.amazon.awssdk.services.dynamodb.model.ScalarAttributeType;
import software.amazon.awssdk.services.dynamodb.model.ScalarAttributeType;
import software.amazon.awssdk.services.dynamodb.model.KeySchemaElement;
import software.amazon.awssdk.services.dynamodb.model.ProvisionedThroughput;
import software.amazon.awssdk.services.dynamodb.model.KeyType;
import software.amazon.awssdk.services.dynamodb.model.CreateTableResponse;
import software.amazon.awssdk.services.dynamodb.model.DescribeTableRequest;
import software.amazon.awssdk.services.dynamodb.model.DescribeTableResponse;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.waiters.DynamoDbWaiter;
```

## Codice

```
public static String createTable(DynamoDbClient ddb, String tableName, String key)
{

    DynamoDbWaiter dbWaiter = ddb.waiter();
    CreateTableRequest request = CreateTableRequest.builder()
        .attributeDefinitions(AttributeDefinition.builder()
            .attributeName(key)
            .attributeType(ScalarAttributeType.S)
            .build())
        .keySchema(KeySchemaElement.builder()
            .attributeName(key)
            .keyType(KeyType.HASH)
            .build())
        .provisionedThroughput(ProvisionedThroughput.builder()
            .readCapacityUnits(new Long(10))
            .writeCapacityUnits(new Long(10))
            .build())
        .tableName(tableName)
        .build();

    String newTable = "";
    try {
        CreateTableResponse response = ddb.createTable(request);
        DescribeTableRequest tableRequest = DescribeTableRequest.builder()
            .tableName(tableName)
```

```

        .build();

        // Wait until the Amazon DynamoDB table is created
        WaiterResponse<DescribeTableResponse> waiterResponse =
dbWaiter.waitForTableExists(tableRequest);
        waiterResponse.matched().response().ifPresent(System.out::println);

        newTable = response.tableDescription().tableName();
        return newTable;

    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return "";
}

```

Vedi l'[esempio completo](#) su GitHub

## Creazione di una tabella con una chiave primaria composta

L'esempio seguente crea una tabella con due attributi. Entrambi gli attributi vengono utilizzati per la chiave primaria composta.

### Importazioni

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.AttributeDefinition;
import software.amazon.awssdk.services.dynamodb.model.CreateTableRequest;
import software.amazon.awssdk.services.dynamodb.model.CreateTableResponse;
import software.amazon.awssdk.services.dynamodb.model.KeySchemaElement;
import software.amazon.awssdk.services.dynamodb.model.KeyType;
import software.amazon.awssdk.services.dynamodb.model.ProvisionedThroughput;
import software.amazon.awssdk.services.dynamodb.model.ScalarAttributeType;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;

```

### Codice

```

public static String createTableComKey(DynamoDbClient ddb, String tableName) {
    CreateTableRequest request = CreateTableRequest.builder()
        .attributeDefinitions(

```

```

        AttributeDefinition.builder()
            .attributeName("Language")
            .attributeType(ScalarAttributeType.S)
            .build(),
        AttributeDefinition.builder()
            .attributeName("Greeting")
            .attributeType(ScalarAttributeType.S)
            .build()
    ).keySchema(
        KeySchemaElement.builder()
            .attributeName("Language")
            .keyType(KeyType.HASH)
            .build(),
        KeySchemaElement.builder()
            .attributeName("Greeting")
            .keyType(KeyType.RANGE)
            .build()
    ).provisionedThroughput(
        ProvisionedThroughput.builder()
            .readCapacityUnits(new Long(10))
            .writeCapacityUnits(new Long(10)).build()
    ).tableName(tableName)
    .build();

String tableId = "";

try {
    CreateTableResponse result = ddb.createTable(request);
    tableId = result.tableDescription().tableId();
    return tableId;
} catch (DynamoDbException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
return "";
}

```

Vedi [l'esempio completo](#) su GitHub.

## Elencare tabelle

È possibile elencare le tabelle in una particolare regione chiamando il `DynamoDbClient`'s `listTables` metodo.

**Note**

Se la tabella denominata non esiste per il tuo account e la tua regione, [ResourceNotFoundException](#) viene generata a.

**Importazioni**

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.ListTablesResponse;
import software.amazon.awssdk.services.dynamodb.model.ListTablesRequest;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import java.util.List;
```

**Codice**

```
public static void listAllTables(DynamoDbClient ddb){

    boolean moreTables = true;
    String lastName = null;

    while(moreTables) {
        try {
            ListTablesResponse response = null;
            if (lastName == null) {
                ListTablesRequest request = ListTablesRequest.builder().build();
                response = ddb.listTables(request);
            } else {
                ListTablesRequest request = ListTablesRequest.builder()
                    .exclusiveStartTableName(lastName).build();
                response = ddb.listTables(request);
            }

            List<String> tableNames = response.tableNames();

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

```
        System.exit(0);
    }

    lastName = response.lastEvaluatedTableName();
    if (lastName == null) {
        moreTables = false;
    }
} catch (DynamoDbException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
}
System.out.println("\nDone!");
}
```

Per impostazione predefinita, vengono restituite fino a 100 tabelle per chiamata: da utilizzare `lastEvaluatedTableName` sull'[ListTablesResponse](#) oggetto restituito per ottenere l'ultima tabella valutata. Puoi utilizzare questo valore per avviare la visualizzazione dell'elenco dopo l'ultimo valore restituito dalla visualizzazione dell'elenco precedente.

Vedi l'esempio [completo](#) su GitHub

## Descrizione di (recupero delle informazioni su) una tabella

Utilizzate il `DynamoDbClient`'s `describeTable` metodo per ottenere informazioni su una tabella.

### Note

Se la tabella denominata non esiste per il tuo account e la tua regione, [ResourceNotFoundException](#) viene generata a.

## Importazioni

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeDefinition;
import software.amazon.awssdk.services.dynamodb.model.DescribeTableRequest;
import software.amazon.awssdk.services.dynamodb.model.ProvisionedThroughputDescription;
import software.amazon.awssdk.services.dynamodb.model.TableDescription;
import java.util.List;
```



## Codice

```
public static void describeDynamoDBTable(DynamoDbClient ddb,String tableName ) {

    DescribeTableRequest request = DescribeTableRequest.builder()
        .tableName(tableName)
        .build();

    try {
        TableDescription tableInfo =
            ddb.describeTable(request).table();

        if (tableInfo != null) {
            System.out.format("Table name   : %s\n",
                tableInfo.tableName());
            System.out.format("Table ARN   : %s\n",
                tableInfo.tableArn());
            System.out.format("Status      : %s\n",
                tableInfo.tableStatus());
            System.out.format("Item count  : %d\n",
                tableInfo.itemCount().longValue());
            System.out.format("Size (bytes): %d\n",
                tableInfo.tableSizeBytes().longValue());

            ProvisionedThroughputDescription throughputInfo =
                tableInfo.provisionedThroughput();
            System.out.println("Throughput");
            System.out.format("  Read Capacity : %d\n",
                throughputInfo.readCapacityUnits().longValue());
            System.out.format("  Write Capacity: %d\n",
                throughputInfo.writeCapacityUnits().longValue());

            List<AttributeDefinition> attributes =
                tableInfo.attributeDefinitions();
            System.out.println("Attributes");

            for (AttributeDefinition a : attributes) {
                System.out.format("  %s (%s)\n",
                    a.attributeName(), a.attributeType());
            }
        }
    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

```
    }  
    System.out.println("\nDone!");  
}
```

Vedi l'[esempio completo](#) su GitHub

## Modifica (aggiornamento) di una tabella

È possibile modificare i valori di throughput assegnati alla tabella in qualsiasi momento chiamando il `DynamoDbClient`'s `updateTable` metodo.

### Note

Se la tabella denominata non esiste per il tuo account e la tua regione, [ResourceNotFoundException](#) viene generata a.

## Importazioni

```
import software.amazon.awssdk.regions.Region;  
import software.amazon.awssdk.services.dynamodb.model.ProvisionedThroughput;  
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;  
import software.amazon.awssdk.services.dynamodb.model.UpdateTableRequest;  
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
```

## Codice

```
public static void updateDynamoDBTable(DynamoDbClient ddb,  
                                       String tableName,  
                                       Long readCapacity,  
                                       Long writeCapacity) {  
  
    System.out.format(  
        "Updating %s with new provisioned throughput values\n",  
        tableName);  
    System.out.format("Read capacity : %d\n", readCapacity);  
    System.out.format("Write capacity : %d\n", writeCapacity);  
  
    ProvisionedThroughput tableThroughput = ProvisionedThroughput.builder()  
        .readCapacityUnits(readCapacity)  
        .writeCapacityUnits(writeCapacity)
```

```
        .build();

        UpdateTableRequest request = UpdateTableRequest.builder()
            .provisionedThroughput(tableThroughput)
            .tableName(tableName)
            .build();

        try {
            ddb.updateTable(request);
        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }

        System.out.println("Done!");
    }
}
```

Vedi l'[esempio completo](#) su GitHub

## Eliminazione di una tabella

Per eliminare una tabella, chiama `DynamoDbClient`'s `deleteTable` method e fornisci il nome della tabella.

### Note

Se la tabella denominata non esiste per il tuo account e la tua regione, [ResourceNotFoundException](#) viene generata a.

## Importazioni

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.DeleteTableRequest;
```

## Codice

```
public static void deleteDynamoDBTable(DynamoDbClient ddb, String tableName) {
```

```
DeleteTableRequest request = DeleteTableRequest.builder()
    .tableName(tableName)
    .build();

try {
    ddb.deleteTable(request);
} catch (DynamoDbException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
System.out.println(tableName + " was successfully deleted!");
}
```

Vedi l'[esempio completo](#) su GitHub

## Ulteriori informazioni

- [Linee guida per l'utilizzo delle tabelle](#) nella Guida per Amazon DynamoDB gli sviluppatori
- [Utilizzo delle tabelle DynamoDB nella](#) Guida per gli Amazon DynamoDB sviluppatori

## Lavora con gli elementi in DynamoDB

In DynamoDB, una voce è una raccolta di attributi, ognuno dei quali dispone di un nome e un valore. Un valore attributo può essere un tipo scalare, set o documento. Per ulteriori informazioni, consulta [Regole di denominazione e tipi di dati nella Guida](#) per gli Amazon DynamoDB sviluppatori.

### Recuperare (ottenere) un item da una tabella

Chiama il `getItem` metodo `DynamoDbClient`'s e passagli un [GetItemRequest](#) oggetto con il nome della tabella e il valore della chiave primaria dell'elemento desiderato. Restituisce un [GetItemResponse](#) oggetto con tutti gli attributi di quell'elemento. Puoi specificare una o più [espressioni di proiezione](#) in `GetItemRequest` per recuperare attributi specifici.

È possibile utilizzare il `item()` metodo dell'`GetItemResponse` oggetto restituito per recuperare una [mappa](#) di coppie chiave (String [AttributeValue](#)) e valore () associate all'elemento.

### Importazioni

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
```

```
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.GetItemRequest;
import java.util.HashMap;
import java.util.Map;
import java.util.Set;
```

## Codice

```
public static void getDynamoDBItem(DynamoDbClient ddb,String tableName,String
key,String keyVal ) {

    HashMap<String,AttributeValue> keyToGet = new HashMap<String,AttributeValue>();

    keyToGet.put(key, AttributeValue.builder()
        .s(keyVal).build());

    GetItemRequest request = GetItemRequest.builder()
        .key(keyToGet)
        .tableName(tableName)
        .build();

    try {
        Map<String,AttributeValue> returnedItem = ddb.getItem(request).item();

        if (returnedItem != null) {
            Set<String> keys = returnedItem.keySet();
            System.out.println("Amazon DynamoDB table attributes: \n");

            for (String key1 : keys) {
                System.out.format("%s: %s\n", key1,
returnedItem.get(key1).toString());
            }
        } else {
            System.out.format("No item found with the key %s!\n", key);
        }
    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

Vedi l'[esempio completo](#) su GitHub

## Recupero (Get) di una voce da una tabella utilizzando il client asincrono

Invocate il `getItem` metodo di `DynamoDbAsyncClient` e passategli un [GetItemRequest](#) oggetto con il nome della tabella e il valore della chiave primaria dell'elemento desiderato.

È possibile restituire un'istanza di [raccolta](#) con tutti gli attributi per tale voce (fare riferimento all'esempio seguente).

### Importazioni

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.model.GetItemRequest;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.DynamoDbAsyncClient;
import java.util.HashMap;
import java.util.Map;
import java.util.Set;
import java.util.stream.Collectors;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
```

### Codice

```
public static void getItem(DynamoDbAsyncClient client, String tableName, String
key, String keyVal) {

    HashMap<String, AttributeValue> keyToGet =
        new HashMap<String, AttributeValue>();

    keyToGet.put(key, AttributeValue.builder()
        .s(keyVal).build());

    try {

        // Create a GetItemRequest instance
        GetItemRequest request = GetItemRequest.builder()
            .key(keyToGet)
            .tableName(tableName)
            .build();

        // Invoke the DynamoDbAsyncClient object's getItem
        java.util.Collection<AttributeValue> returnedItem =
            client.getItem(request).join().item().values();
```

```
        // Convert Set to Map
        Map<String, AttributeValue> map =
returnedItem.stream().collect(Collectors.toMap(AttributeValue::s, s->s));
        Set<String> keys = map.keySet();
        for (String sinKey : keys) {
            System.out.format("%s: %s\n", sinKey, map.get(sinKey).toString());
        }

    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

Vedi l'[esempio completo](#) su GitHub

## Aggiunta di un nuovo item a una tabella

Creare una [mappa](#) di coppie chiave-valore che rappresentino gli attributi della voce. Queste devono includere valori per i campi chiave primaria della tabella. Se l'elemento identificato dalla chiave primaria esiste già, i relativi campi vengono aggiornati dalla richiesta.

### Note

Se la tabella denominata non esiste per il tuo account e la tua regione, [ResourceNotFoundException](#) viene generata a.

## Importazioni

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.PutItemRequest;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;
import java.util.HashMap;
```

## Codice

```
public static void putItemInTable(DynamoDbClient ddb,
```

```
        String tableName,
        String key,
        String keyVal,
        String albumTitle,
        String albumTitleValue,
        String awards,
        String awardVal,
        String songTitle,
        String songTitleVal){

    HashMap<String,AttributeValue> itemValues = new
HashMap<String,AttributeValue>();

    // Add all content to the table
    itemValues.put(key, AttributeValue.builder().s(keyVal).build());
    itemValues.put(songTitle, AttributeValue.builder().s(songTitleVal).build());
    itemValues.put(albumTitle,
AttributeValue.builder().s(albumTitleValue).build());
    itemValues.put(awards, AttributeValue.builder().s(awardVal).build());

    PutItemRequest request = PutItemRequest.builder()
        .tableName(tableName)
        .item(itemValues)
        .build();

    try {
        ddb.putItem(request);
        System.out.println(tableName + " was successfully updated");

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

Vedi l'[esempio completo](#) su GitHub



## Aggiornamento di un item esistente in una tabella

È possibile aggiornare un attributo per una voce che esiste già in una tabella utilizzando il metodo `updateItem` di `DynamoDbClient`, fornendo un nome di tabella, un valore chiave primaria e una mappa di campi da aggiornare.

### Note

Se la tabella denominata non esiste per il tuo account e la tua regione, o se l'elemento identificato dalla chiave primaria che hai passato non esiste, [ResourceNotFoundException](#) viene generato a.

## Importazioni

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.AttributeAction;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.AttributeValueUpdate;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemRequest;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import java.util.HashMap;
```

## Codice

```
public static void updateTableItem(DynamoDbClient ddb,
                                   String tableName,
                                   String key,
                                   String keyVal,
                                   String name,
                                   String updateVal){

    HashMap<String,AttributeValue> itemKey = new HashMap<String,AttributeValue>();

    itemKey.put(key, AttributeValue.builder().s(keyVal).build());

    HashMap<String,AttributeValueUpdate> updatedValues =
        new HashMap<String,AttributeValueUpdate>();
```

```
// Update the column specified by name with updatedVal
updatedValues.put(name, AttributeValueUpdate.builder()
    .value(AttributeValue.builder().s(updateVal).build())
    .action(AttributeAction.PUT)
    .build());

UpdateItemRequest request = UpdateItemRequest.builder()
    .tableName(tableName)
    .key(itemKey)
    .attributeUpdates(updatedValues)
    .build();

try {
    ddb.updateItem(request);
} catch (ResourceNotFoundException e) {
    System.err.println(e.getMessage());
    System.exit(1);
} catch (DynamoDbException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}

System.out.println("Done!");
}
```

Vedi l'[esempio completo](#) su GitHub

## Aggiornamento di una voce (item) esistente in una tabella

È possibile eliminare un elemento esistente in una tabella utilizzando il `deleteItem` metodo `DynamoDbClient`'s e fornendo un nome di tabella oltre al valore della chiave primaria.

### Note

Se la tabella denominata non esiste per il tuo account e la tua regione, o se l'elemento identificato dalla chiave primaria che hai passato non esiste, [ResourceNotFoundException](#) viene generato un.

## Importazioni

```
import software.amazon.awssdk.regions.Region;
```

```
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DeleteItemRequest;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import java.util.HashMap;
```

## Codice

```
public static void deleteDynamoDBItem(DynamoDbClient ddb, String tableName, String
key, String keyVal) {

    HashMap<String,AttributeValue> keyToGet =
        new HashMap<String,AttributeValue>();

    keyToGet.put(key, AttributeValue.builder()
        .s(keyVal)
        .build());

    DeleteItemRequest deleteReq = DeleteItemRequest.builder()
        .tableName(tableName)
        .key(keyToGet)
        .build();

    try {
        ddb.deleteItem(deleteReq);
    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

Vedi l'[esempio completo](#) su. GitHub

## Ulteriori informazioni

- [Linee guida per l'utilizzo degli elementi](#) nella Guida per gli Amazon DynamoDB sviluppatori
- [Utilizzo degli elementi della DynamoDB](#) Guida per gli Amazon DynamoDB sviluppatori

## Mappa gli oggetti Java sugli elementi DynamoDB con AWS SDK for Java 2.x

L'API [DynamoDB Enhanced Client](#) è una libreria di alto livello che è il successore della classe di SDK for DynamoDBMapper Java v1.x. Offre un modo semplice per mappare le classi lato client alle tabelle DynamoDB. Le relazioni tra le tabelle e le classi di dati corrispondenti vengono definite nel codice. Dopo aver definito queste relazioni, è possibile eseguire in modo intuitivo varie operazioni di creazione, lettura, aggiornamento o eliminazione (CRUD) su tabelle o elementi in DynamoDB.

L'API Enhanced Client di DynamoDB include anche [l'API Enhanced Document](#) che consente di lavorare con elementi di tipo documento che non seguono uno schema definito.

L'API DynamoDB Enhanced Client viene discussa nei seguenti argomenti.

- [Inizia a usare l'API DynamoDB Enhanced Client](#)
- [Scopri le basi dell'API DynamoDB Enhanced Client](#)
- [Usa funzionalità di mappatura avanzate](#)
- [Lavora con documenti JSON con l'API Enhanced Document per DynamoDB](#)
- [Usa le estensioni](#)
- [Usa l'API DynamoDB Enhanced Client in modo asincrono](#)
- [Annotazioni di classi di dati](#)

### Inizia a usare l'API DynamoDB Enhanced Client

Il seguente tutorial presenta le nozioni di base necessarie per lavorare con l'API DynamoDB Enhanced Client.

#### Aggiungi dipendenze

Per iniziare a utilizzare l'API DynamoDB Enhanced Client nel tuo progetto, aggiungi una dipendenza dall'artefatto Maven. dynamodb-enhanced. Questo è illustrato negli esempi seguenti.

#### Maven

```
<project>
  <dependencyManagement>
    <dependencies>
```

```
<dependency>
  <groupId>software.amazon.awssdk</groupId>
  <artifactId>bom</artifactId>
  <version><VERSION></version>
  <type>pom</type>
  <scope>import</scope>
</dependency>
</dependencies>
</dependencyManagement>
<dependencies>
  <dependency>
    <groupId>software.amazon.awssdk</groupId>
    <artifactId>dynamodb-enhanced</artifactId>
  </dependency>
</dependencies>
...
</project>
```

Esegui una ricerca nell'archivio centrale di Maven per la [versione più recente](#) e sostituiscila <VERSION> con questo valore.

## Gradle

```
repositories {
  mavenCentral()
}
dependencies {
  implementation(platform("software.amazon.awssdk:bom:<VERSION>"))
  implementation("software.amazon.awssdk:dynamodb-enhanced")
  ...
}
```

<VERSION> Esegui una ricerca nell'archivio centrale di Maven per la [versione più recente](#) e sostituiscila con questo valore.

## Genera un **TableSchema** da una classe di dati

A [TableSchema](#) consente al client avanzato di mappare i valori degli attributi DynamoDB da e verso le classi lato client. In questo tutorial, imparerai a conoscere TableSchema i file derivati da una classe di dati statici e generati dal codice utilizzando un builder.

## Usa una classe di dati annotata

L'SDK for Java 2.x include [un set di](#) annotazioni che è possibile utilizzare con una classe di dati per generare `TableSchema` rapidamente un file per mappare le classi alle tabelle.

[Inizia creando una classe di dati conforme alle specifiche. `JavaBean`](#) La specifica richiede che una classe disponga di un costruttore pubblico senza argomenti e che disponga di getter e setter per ogni attributo della classe. Includi un'annotazione a livello di classe per indicare che la classe di dati è una `DynamoDbBean`. Inoltre, includi almeno un'`DynamoDbPartitionKey` annotazione sul getter o setter per l'attributo chiave primaria.

È possibile applicare [annotazioni a livello di attributo a](#) getter o setter, ma non a entrambi.

### Note

Il termine `property` viene normalmente utilizzato per un valore incapsulato in un `JavaBean`. Tuttavia, questa guida utilizza `attribute` invece il termine, per coerenza con la terminologia utilizzata da `DynamoDB`.

La `Customer` classe seguente mostra le annotazioni che collegano la definizione della classe a una tabella `DynamoDB`.

### Classe `Customer`

```
package org.example.tests.model;

import software.amazon.awssdk.enhanced.dynamodb.mapper.annotations.DynamoDbBean;
import
    software.amazon.awssdk.enhanced.dynamodb.mapper.annotations.DynamoDbPartitionKey;
import software.amazon.awssdk.enhanced.dynamodb.mapper.annotations.DynamoDbSortKey;

import java.time.Instant;

@DynamoDbBean
public class Customer {

    private String id;
    private String name;
    private String email;
    private Instant regDate;
```

```
@DynamoDbPartitionKey
public String getId() { return this.id; }

public void setId(String id) { this.id = id; }

public String getCustName() { return this.name; }

public void setCustName(String name) { this.name = name; }

@DynamoDbSortKey
public String getEmail() { return this.email; }

public void setEmail(String email) { this.email = email; }

public Instant getRegistrationDate() { return this.regDate; }

public void setRegistrationDate(Instant registrationDate) { this.regDate =
registrationDate; }

@Override
public String toString() {
    return "Customer [id=" + id + ", name=" + name + ", email=" + email
        + ", regDate=" + regDate + "]";
}
}
```

Dopo aver creato una classe di dati annotata, usatela per creare la `TableSchema`, come mostrato nel seguente frammento.

```
static final TableSchema<Customer> customerTableSchema =
    TableSchema.fromBean(Customer.class);
```

A `TableSchema` è progettato per essere statico e immutabile. Di solito puoi istanziarlo al momento del caricamento della classe.

Il metodo statico `TableSchema.fromBean()` factory analizza il bean per generare la mappatura degli attributi (proprietà) delle classi di dati da e verso gli attributi DynamoDB.

Per un esempio di utilizzo di un modello di dati composto da diverse classi di dati, consultate la classe nella `Person` sezione. [???](#)

## Usa un builder

Puoi saltare il costo dell'introspezione dei bean se definisci lo schema della tabella nel codice. Se codifichi lo schema, la tua classe non ha bisogno di seguire gli standard di JavaBean denominazione né di essere annotata. L'esempio seguente utilizza un generatore ed è equivalente all'esempio di `Customer` classe che utilizza le annotazioni.

```
static final TableSchema<Customer> customerTableSchema =
    TableSchema.builder(Customer.class)
        .newItemSupplier(Customer::new)
        .addAttribute(String.class, a -> a.name("id")
            .getter(Customer::getId)
            .setter(Customer::setId)
            .tags(StaticAttributeTags.primaryPartitionKey()))
        .addAttribute(String.class, a -> a.name("email")
            .getter(Customer::getEmail)
            .setter(Customer::setEmail)
            .tags(StaticAttributeTags.primarySortKey()))
        .addAttribute(String.class, a -> a.name("name")
            .getter(Customer::getCustName)
            .setter(Customer::setCustName))
        .addAttribute(Instant.class, a -> a.name("registrationDate")
            .getter(Customer::getRegistrationDate)
            .setter(Customer::setRegistrationDate))
        .build();
```

## Crea un client avanzato e **DynamoDbTable**

### Crea un client avanzato

La [DynamoDbEnhancedClient](#) classe, o la sua controparte asincrona [DynamoDbEnhancedAsyncClient](#), è il punto di partenza per lavorare con l'API DynamoDB Enhanced Client.

Il client avanzato richiede uno standard per eseguire il lavoro. [DynamoDbClient](#) L'API offre due modi per creare un `DynamoDbEnhancedClient` istanza. La prima opzione, mostrata nel frammento seguente, crea uno standard `DynamoDbClient` con le impostazioni predefinite raccolte dalle impostazioni di configurazione.

```
DynamoDbEnhancedClient enhancedClient = DynamoDbEnhancedClient.create();
```



Se desideri configurare il client standard sottostante, puoi fornirlo al metodo builder del client avanzato, come mostrato nel seguente frammento.

```
// Configure an instance of the standard DynamoDbClient.
DynamoDbClient standardClient = DynamoDbClient.builder()
    .region(Region.US_EAST_1)
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

// Use the configured standard client with the enhanced client.
DynamoDbEnhancedClient enhancedClient = DynamoDbEnhancedClient.builder()
    .dynamoDbClient(standardClient)
    .build();
```

## Creazione di un'istanza **DynamoDbTable**

Pensate a [DynamoDbTable](#) come alla rappresentazione lato client di una tabella DynamoDB che utilizza la funzionalità di mappatura fornita da `TableSchema`. La `DynamoDbTable` classe fornisce metodi per le operazioni CRUD che consentono di interagire con una singola tabella DynamoDB.

`DynamoDbTable<T>` è una classe generica che accetta un argomento di tipo singolo, sia che si tratti di una classe personalizzata o di un argomento `EnhancedDocument` quando si lavora con elementi di tipo documento. Questo tipo di argomento stabilisce la relazione tra la classe utilizzata e la singola tabella DynamoDB.

Utilizzate il metodo `table()` factory di `DynamoDbEnhancedClient` per creare un'`DynamoDbTable` istanza, come mostrato nel frammento seguente.

```
static final DynamoDbTable<Customer> customerTable =
    enhancedClient.table("Customer", TableSchema.fromBean(Customer.class));
```

`DynamoDbTable` le istanze sono candidate per essere utilizzate singleton perché sono immutabili e possono essere utilizzate in tutta l'applicazione.

Il codice ora ha una rappresentazione in memoria di una tabella DynamoDB che può funzionare con le istanze. `Customer` La tabella DynamoDB effettiva potrebbe esistere o meno. Se la tabella denominata `Customer` esiste già, puoi iniziare a eseguire operazioni CRUD su di essa. Se non esiste, utilizzate l'`DynamoDbTable` istanza per creare la tabella come illustrato nella sezione successiva.

## Se necessario, creare una tabella DynamoDB

Dopo aver creato un `DynamoDbTable` istanza, utilizzala per creare una sola volta una tabella in DynamoDB.

Crea un codice di esempio per la tabella

L'esempio seguente crea una tabella DynamoDB basata sulla `Customer` classe di dati.

Questo esempio crea una tabella DynamoDB con il `Customer` nome, identico al nome della classe, ma il nome della tabella può essere qualcos'altro. Qualunque sia il nome dato alla tabella, è necessario utilizzare questo nome in altre applicazioni per lavorare con la tabella. Fornite questo nome al `table()` metodo ogni volta che create un altro `DynamoDbTable` oggetto per lavorare con la tabella DynamoDB sottostante.

[Il parametro Java `lambdabuilder`, passato al `createTable` metodo consente di personalizzare la tabella.](#) In questo esempio, viene configurato il [throughput assegnato](#). Se desideri utilizzare le impostazioni predefinite quando crei una tabella, salta il generatore, come mostrato nel frammento seguente.

```
customerTable.createTable();
```

Quando vengono utilizzate le impostazioni predefinite, i valori per il throughput assegnato non vengono impostati. [Al contrario, la modalità di fatturazione per la tabella è impostata su richiesta.](#)

L'esempio utilizza anche un [DynamoDbWaiter](#) prima di tentare di stampare il nome della tabella ricevuto nella risposta. La creazione di una tabella richiede del tempo. Pertanto, l'utilizzo di un cameriere significa che non è necessario scrivere una logica che interroghi il servizio DynamoDB per verificare se la tabella esiste prima di utilizzarla.

## Importazioni

```
import com.example.dynamodb.Customer;
import software.amazon.awssdk.core.internal.waiters.ResponseOrException;
import software.amazon.awssdk.enhanced.dynamodb.DynamoDbEnhancedClient;
import software.amazon.awssdk.enhanced.dynamodb.DynamoDbTable;
import software.amazon.awssdk.enhanced.dynamodb.TableSchema;
import software.amazon.awssdk.services.dynamodb.model.DescribeTableResponse;
import software.amazon.awssdk.services.dynamodb.waiters.DynamoDbWaiter;
```

## Codice

```
public static void createCustomerTable(DynamoDbTable<Customer> customerTable,
DynamoDbClient standardClient) {
    // Create the DynamoDB table using the 'customerTable' DynamoDbTable instance.
    customerTable.createTable(builder -> builder
        .provisionedThroughput(b -> b
            .readCapacityUnits(10L)
            .writeCapacityUnits(10L)
            .build())
    );
    // The DynamoDbClient instance (named 'standardClient') passed to the builder for
the DynamoDbWaiter is the same instance
    // that was passed to the builder of the DynamoDbEnhancedClient instance that we
created previously.
    // By using the same instance, it ensures that the same Region that was configured
on the standard DynamoDbClient
    // instance is used for other service clients that accept a DynamoDbClient during
construction.
    try (DynamoDbWaiter waiter =
DynamoDbWaiter.builder().client(standardClient).build()) { // DynamoDbWaiter is
Autocloseable
        ResponseOrException<DescribeTableResponse> response = waiter
            .waitUntilTableExists(builder ->
builder.tableName("Customer").build())
            .matched();
        DescribeTableResponse tableDescription = response.response().orElseThrow(
            () -> new RuntimeException("Customer table was not created."));
        // The actual error can be inspected in response.exception()
        logger.info("Customer table was created.");
    }
}
```

### Note

I nomi degli attributi di una tabella DynamoDB iniziano con una lettera minuscola quando la tabella viene generata da una classe di dati. Se desideri che il nome dell'attributo della tabella inizi con una lettera maiuscola, usa l'[@DynamoDbAttribute\(NAME\)annotazione](#) e fornisci il nome che desideri come parametro.

## Esegui operazioni

Dopo aver creato la tabella, utilizzate l'`DynamoDbTable`istanza per eseguire operazioni sulla tabella DynamoDB.

Nell'esempio seguente, un singleton `DynamoDbTable<Customer>` viene passato come parametro insieme a un'istanza della [classe di Customer dati](#) per aggiungere un nuovo elemento alla tabella.

```
public static void putItemExample(DynamoDbTable<Customer> customerTable, Customer
customer){
    logger.info(customer.toString());
    customerTable.putItem(customer);
}
```

## Oggetto **Customer**

```
Customer customer = new Customer();
customer.setId("1");
customer.setCustName("Customer Name");
customer.setEmail("customer@example.com");
customer.setRegistrationDate(Instant.parse("2023-07-03T10:15:30.00Z"));
```

Prima di inviare l'`customer` oggetto al servizio DynamoDB, registrate l'output del metodo `toString()` dell'oggetto per confrontarlo con quello inviato dal client avanzato.

```
Customer [id=1, name=Customer Name, email=customer@example.com,
regDate=2023-07-03T10:15:30Z]
```

La registrazione a livello di cavo mostra il payload della richiesta generata. Il client avanzato ha generato la rappresentazione di basso livello dalla classe di dati. L'`regDate` attributo, che è un `Instant` tipo in Java, è rappresentato come una stringa DynamoDB.

```
{
  "TableName": "Customer",
  "Item": {
    "registrationDate": {
      "S": "2023-07-03T10:15:30Z"
    },
    "id": {
      "S": "1"
    }
  }
}
```

```
    },
    "custName": {
        "S": "Customer Name"
    },
    "email": {
        "S": "customer@example.com"
    }
}
}
```

## Lavora con una tabella esistente

La sezione precedente mostrava come creare una tabella DynamoDB a partire da una classe di dati Java. Se disponi già di una tabella esistente e desideri utilizzare le funzionalità del client avanzato, puoi creare una classe di dati Java per lavorare con la tabella. È necessario esaminare la tabella DynamoDB e aggiungere le annotazioni necessarie alla classe di dati.

Prima di lavorare con una tabella esistente, chiamate il metodo `DynamoDbEnhanced.table()`. Ciò è stato fatto nell'esempio precedente con la seguente dichiarazione.

```
DynamoDbTable<Customer> customerTable = enhancedClient.table("Customer",
    TableSchema.fromBean(Customer.class));
```

Dopo la restituzione dell'`DynamoDbTable` istanza, potete iniziare subito a lavorare con la tabella sottostante. Non è necessario ricreare la tabella chiamando il `DynamoDbTable.createTable()` metodo.

L'esempio seguente lo dimostra recuperando immediatamente un'`Customer` istanza dalla tabella DynamoDB.

```
DynamoDbTable<Customer> customerTable = enhancedClient.table("Customer",
    TableSchema.fromBean(Customer.class));
// The Customer table exists already and has an item with a primary key value of "1"
// and a sort key value of "customer@example.com".
customerTable.getItem(
    Key.builder()
        .partitionValue("1").
        sortValue("customer@example.com").build());
```

**⚠ Important**

Il nome della tabella utilizzato nel `table()` metodo deve corrispondere al nome della tabella DynamoDB esistente.

## Scopri le basi dell'API DynamoDB Enhanced Client

[Questo argomento illustra le funzionalità di base dell'API DynamoDB Enhanced Client e la confronta con l'API client DynamoDB standard.](#)

Se non conosci l'API DynamoDB Enhanced Client, ti consigliamo di seguire il tutorial [introduttivo per familiarizzare con le](#) classi fondamentali.

### Elementi DynamoDB in Java

Le tabelle DynamoDB memorizzano gli elementi. A seconda del caso d'uso, gli elementi sul lato Java possono assumere la forma di dati strutturati staticamente o di strutture create dinamicamente.

Se il tuo caso d'uso richiede elementi con un set coerente di attributi, usa [classi annotate](#) o usa un [builder](#) per generare i tipi statici appropriati. `TableSchema`

In alternativa, se devi archiviare elementi costituiti da strutture diverse, crea un.

`DocumentTableSchema` `DocumentTableSchema` fa parte dell'[API Enhanced Document](#) e richiede solo una chiave primaria tipizzata staticamente e funziona con le `EnhancedDocument` istanze per contenere gli elementi di dati. [L'API Enhanced Document è trattata in un altro argomento.](#)

### Tipi di attributi per le classi del modello di dati

Sebbene DynamoDB [supporti un numero limitato di tipi di attributi rispetto al rich type system di](#) Java, l'API DynamoDB Enhanced Client fornisce meccanismi per convertire i membri di una classe Java da e verso i tipi di attributi DynamoDB.

I tipi di attributi (proprietà) delle classi di dati Java devono essere tipi di oggetti, non primitivi. Ad esempio, utilizzate `Long` sempre tipi di dati `Integer` oggetto, non `long` e `int` primitivi.

[Per impostazione predefinita, l'API DynamoDB Enhanced Client supporta convertitori di attributi per un gran numero di tipi, come Integer, String e Instant. BigDecimal](#) L'elenco viene visualizzato nelle [classi di implementazione note](#) dell'interfaccia. `AttributeConverter` L'elenco include molti tipi e raccolte come mappe, elenchi e set.

Per memorizzare i dati per un tipo di attributo che non è supportato di default o non è conforme alla JavaBean convenzione, puoi scrivere un'AttributeConverter implementazione personalizzata per eseguire la conversione. Per un [esempio](#), consulta la sezione sulla conversione degli attributi.

Per memorizzare i dati per un tipo di attributo la cui classe è conforme alla specifica Java beans (o una [classe di dati immutabile](#)), puoi adottare due approcci.

- Se avete accesso al file sorgente, potete annotare la classe con `@DynamoDbBean` (o) `@DynamoDbImmutable`. La sezione che illustra gli attributi annidati mostra [esempi](#) di utilizzo di classi annotate.
- Se non hai accesso al file sorgente della classe di JavaBean dati per l'attributo (o non vuoi annotare il file sorgente di una classe a cui hai accesso), puoi usare l'approccio builder. Questo crea uno schema tabellare senza definire le chiavi. Quindi, puoi annidare questo schema di tabella all'interno di un altro schema di tabella per eseguire la mappatura. La sezione degli attributi nidificati contiene un [esempio](#) che mostra l'uso di schemi nidificati.

## Valori nulli

Quando si utilizza l'`putItem` API, il client avanzato non include gli attributi con valori nulli di un oggetto dati mappato nella richiesta a DynamoDB.

Per `updateItem` le richieste, gli attributi con valore nullo vengono rimossi dall'elemento del database. Se intendete aggiornare alcuni valori di attributo e mantenere invariati gli altri, copiate i valori di altri attributi che non devono essere modificati o utilizzate il metodo [ignoreNull\(\)](#) sul generatore di aggiornamenti.

L'esempio seguente mostra il metodo `ignoreNulls()` the `updateItem()`

```
public void updateItemNullsExample(){
    Customer customer = new Customer();
    customer.setCustName("CustName");
    customer.setEmail("email");
    customer.setId("1");
    customer.setRegistrationDate(Instant.now());

    // Put item with values for all attributes.
    customerDynamoDbTable.putItem(customer);
}
```

```
// Create a Customer instance with the same id value, but a different name
value.
// Do not set the 'registrationDate' attribute.
Customer custForUpdate = new Customer();
custForUpdate.setCustName("NewName");
custForUpdate.setEmail("email");
custForUpdate.setId("1");

// Update item without setting the registrationDate attribute.
customerDynamoDbTable.updateItem(b -> b
    .item(custForUpdate)
    .ignoreNulls(Boolean.TRUE));

Customer updatedWithNullsIgnored = customerDynamoDbTable.getItem(customer);
// registrationDate value is unchanged.
logger.info(updatedWithNullsIgnored.toString());

customerDynamoDbTable.updateItem(custForUpdate);
Customer updatedWithNulls = customerDynamoDbTable.getItem(customer);
// registrationDate value is null because ignoreNulls() was not used.
logger.info(updatedWithNulls.toString());
}
}

// Logged lines.
Customer [id=1, custName=NewName, email=email,
  registrationDate=2023-04-05T16:32:32.056Z]
Customer [id=1, custName=NewName, email=email, registrationDate=null]
```

## Metodi di base di DynamoDB Enhanced Client

I metodi di base del client avanzato si associano alle operazioni del servizio DynamoDB da cui prendono il nome. Gli esempi seguenti mostrano la variante più semplice di ciascun metodo. È possibile personalizzare ogni metodo passando un oggetto di richiesta avanzato. Gli oggetti di richiesta avanzati offrono la maggior parte delle funzionalità disponibili nel client DynamoDB standard. Sono completamente documentati nell'API Reference. AWS SDK for Java 2.x

L'esempio utilizza quanto [the section called “Classe Customer”](#) mostrato in precedenza.

```
// CreateTable
customerTable.createTable();

// GetItem
```



```
Customer customer =
    customerTable.getItem(Key.builder().partitionValue("a123").build());

// UpdateItem
Customer updatedCustomer = customerTable.updateItem(customer);

// PutItem
customerTable.putItem(customer);

// DeleteItem
Customer deletedCustomer =
    customerTable.deleteItem(Key.builder().partitionValue("a123").sortValue(456).build());

// Query
PageIterable<Customer> customers = customerTable.query(keyEqualTo(k ->
    k.partitionValue("a123")));

// Scan
PageIterable<Customer> customers = customerTable.scan();

// BatchGetItem
BatchGetResultPageIterable batchResults =
    enhancedClient.batchGetItem(r -> r.addReadBatch(ReadBatch.builder(Customer.class)
        .mappedTableResource(customerTable)
        .addGetItem(key1)
        .addGetItem(key2)
        .addGetItem(key3)
        .build()));

// BatchWriteItem
batchResults = enhancedClient.batchWriteItem(r ->
    r.addWriteBatch(WriteBatch.builder(Customer.class)
        .mappedTableResource(customerTable)
        .addPutItem(customer)
        .addDeleteItem(key1)
        .addDeleteItem(key1)
        .build()));

// TransactGetItems
transactResults = enhancedClient.transactGetItems(r -> r.addGetItem(customerTable,
    key1)
        .addGetItem(customerTable,
    key2));
```

```
// TransactWriteItems
enhancedClient.transactWriteItems(r -> r.addConditionCheck(customerTable,
                                                                    i -> i.key(orderKey)

.conditionExpression(conditionExpression))
                                                                    .addUpdateItem(customerTable, customer)
                                                                    .addDeleteItem(customerTable, key));
```

Confronta DynamoDB Enhanced Client con il client DynamoDB standard

[Entrambe le API client DynamoDB, standard e avanzate, consentono di lavorare con le tabelle DynamoDB per eseguire operazioni CRUD \(creazione, lettura, aggiornamento ed eliminazione\) a livello di dati.](#) La differenza tra le API client sta nel modo in cui ciò viene realizzato. Utilizzando il client standard, lavori direttamente con attributi di dati di basso livello. L'API client avanzata utilizza classi Java familiari e si collega all'API di basso livello dietro le quinte.

Sebbene entrambe le API client supportino operazioni a livello di dati, il client DynamoDB standard supporta anche operazioni a livello di risorsa. Le operazioni a livello di risorsa gestiscono il database, come la creazione di backup, l'elenco delle tabelle e l'aggiornamento delle tabelle. L'API client avanzata supporta un numero selezionato di operazioni a livello di risorsa come la creazione, la descrizione e l'eliminazione di tabelle.

Per illustrare i diversi approcci utilizzati dalle due API client, i seguenti esempi di codice mostrano la creazione della stessa ProductCatalog tabella utilizzando il client standard e il client avanzato.

Confronto: crea una tabella utilizzando il client DynamoDB standard

```
DependencyFactory.dynamoDbClient().createTable(builder -> builder
    .tableName(TABLE_NAME)
    .attributeDefinitions(
        b -> b.attributeName("id").attributeType(ScalarAttributeType.N),
        b -> b.attributeName("title").attributeType(ScalarAttributeType.S),
        b -> b.attributeName("isbn").attributeType(ScalarAttributeType.S)
    )
    .keySchema(
        builder1 -> builder1.attributeName("id").keyType(KeyType.HASH),
        builder2 -> builder2.attributeName("title").keyType(KeyType.RANGE)
    )
    .globalSecondaryIndexes(builder3 -> builder3
        .indexName("products_by_isbn")
        .keySchema(builder2 -> builder2
            .attributeName("isbn").keyType(KeyType.HASH))
```

```

        .projection(builder2 -> builder2
            .projectionType(ProjectionType.INCLUDE)
            .nonKeyAttributes("price", "authors"))
        .provisionedThroughput(builder4 -> builder4
            .writeCapacityUnits(5L).readCapacityUnits(5L))
    )
    .provisionedThroughput(builder1 -> builder1
        .readCapacityUnits(5L).writeCapacityUnits(5L))
);

```

## Confronta: crea una tabella utilizzando il DynamoDB Enhanced Client

```

DynamoDbEnhancedClient enhancedClient = DependencyFactory.enhancedClient();
productCatalog = enhancedClient.table(TABLE_NAME,
    TableSchema.fromImmutableClass(ProductCatalog.class));
productCatalog.createTable(b -> b
    .provisionedThroughput(b1 -> b1.readCapacityUnits(5L).writeCapacityUnits(5L))
    .globalSecondaryIndices(b2 -> b2.indexName("products_by_isbn")
        .projection(b4 -> b4
            .projectionType(ProjectionType.INCLUDE)
            .nonKeyAttributes("price", "authors"))
        .provisionedThroughput(b3 ->
            b3.writeCapacityUnits(5L).readCapacityUnits(5L))
    )
);

```

Il client avanzato utilizza la seguente classe di dati annotati. Il DynamoDB Enhanced Client mappa i tipi di dati Java ai tipi di dati DynamoDB per un codice meno dettagliato e più facile da seguire. `ProductCatalog` è un esempio di utilizzo di una classe immutabile con DynamoDB Enhanced Client. [L'uso delle classi Immutabili per le classi di dati mappate viene discusso più avanti in questo argomento.](#)

## Classe **ProductCatalog**

```

package org.example.tests.model;

import software.amazon.awssdk.enhanced.dynamodb.mapper.annotations.DynamoDbIgnore;
import software.amazon.awssdk.enhanced.dynamodb.mapper.annotations.DynamoDbImmutable;
import
    software.amazon.awssdk.enhanced.dynamodb.mapper.annotations.DynamoDbPartitionKey;
import
    software.amazon.awssdk.enhanced.dynamodb.mapper.annotations.DynamoDbSecondaryPartitionKey;

```

```
import software.amazon.awssdk.enhanced.dynamodb.mapper.annotations.DynamoDbSortKey;

import java.math.BigDecimal;
import java.util.Objects;
import java.util.Set;

@DynamoDbImmutable(builder = ProductCatalog.Builder.class)
public class ProductCatalog implements Comparable<ProductCatalog> {
    private Integer id;
    private String title;
    private String isbn;
    private Set<String> authors;
    private BigDecimal price;

    private ProductCatalog(Builder builder){
        this.authors = builder.authors;
        this.id = builder.id;
        this.isbn = builder.isbn;
        this.price = builder.price;
        this.title = builder.title;
    }

    public static Builder builder(){ return new Builder(); }

    @DynamoDbPartitionKey
    public Integer id() { return id; }

    @DynamoDbSortKey
    public String title() { return title; }

    @DynamoDbSecondaryPartitionKey(indexNames = "products_by_isbn")
    public String isbn() { return isbn; }
    public Set<String> authors() { return authors; }
    public BigDecimal price() { return price; }

    public static final class Builder {
        private Integer id;
        private String title;
        private String isbn;
        private Set<String> authors;
        private BigDecimal price;
        private Builder(){}
```

```
public Builder id(Integer id) { this.id = id; return this; }
public Builder title(String title) { this.title = title; return this; }
public Builder isbn(String ISBN) { this.isbn = ISBN; return this; }
public Builder authors(Set<String> authors) { this.authors = authors; return
this; }
public Builder price(BigDecimal price) { this.price = price; return this; }
public ProductCatalog build() { return new ProductCatalog(this); }
}

@Override
public String toString() {
    final StringBuffer sb = new StringBuffer("ProductCatalog{");
    sb.append("id=").append(id);
    sb.append(", title=").append(title).append('\ ');
    sb.append(", isbn=").append(isbn).append('\ ');
    sb.append(", authors=").append(authors);
    sb.append(", price=").append(price);
    sb.append('}');
    return sb.toString();
}

@Override
public boolean equals(Object o) {
    if (this == o) return true;
    if (o == null || getClass() != o.getClass()) return false;
    ProductCatalog that = (ProductCatalog) o;
    return id.equals(that.id) && title.equals(that.title) && Objects.equals(isbn,
that.isbn) && Objects.equals(authors, that.authors) && Objects.equals(price,
that.price);
}

@Override
public int hashCode() {
    return Objects.hash(id, title, isbn, authors, price);
}

@Override
@DynamoDbIgnore
public int compareTo(ProductCatalog other) {
    if (this.id.compareTo(other.id) != 0){
        return this.id.compareTo(other.id);
    } else {
        return this.title.compareTo(other.title);
    }
}
```

```
    }  
  }  
}
```

I due esempi di codice seguenti di scrittura in batch illustrano la verbosità e la mancanza di sicurezza dei tipi quando si utilizza il client standard anziché il client avanzato.

### Confronto: Scrittura in batch utilizzando il client DynamoDB standard

```
public static void batchWriteStandard(DynamoDbClient dynamoDbClient, String  
tableName) {  
  
    Map<String, AttributeValue> catalogItem = Map.of(  
        "authors", AttributeValue.builder().ss("a", "b").build(),  
        "id", AttributeValue.builder().n("1").build(),  
        "isbn", AttributeValue.builder().s("1-565-85698").build(),  
        "title", AttributeValue.builder().s("Title 1").build(),  
        "price", AttributeValue.builder().n("52.13").build());  
  
    Map<String, AttributeValue> catalogItem2 = Map.of(  
        "authors", AttributeValue.builder().ss("a", "b", "c").build(),  
        "id", AttributeValue.builder().n("2").build(),  
        "isbn", AttributeValue.builder().s("1-208-98073").build(),  
        "title", AttributeValue.builder().s("Title 2").build(),  
        "price", AttributeValue.builder().n("21.99").build());  
  
    Map<String, AttributeValue> catalogItem3 = Map.of(  
        "authors", AttributeValue.builder().ss("g", "k", "c").build(),  
        "id", AttributeValue.builder().n("3").build(),  
        "isbn", AttributeValue.builder().s("7-236-98618").build(),  
        "title", AttributeValue.builder().s("Title 3").build(),  
        "price", AttributeValue.builder().n("42.00").build());  
  
    Set<WriteRequest> writeRequests = Set.of(  
        WriteRequest.builder().putRequest(b -> b.item(catalogItem)).build(),  
        WriteRequest.builder().putRequest(b -> b.item(catalogItem2)).build(),  
        WriteRequest.builder().putRequest(b -> b.item(catalogItem3)).build());  
  
    Map<String, Set<WriteRequest>> productCatalogItems = Map.of(  
        "ProductCatalog", writeRequests);  
  
    BatchWriteItemResponse response = dynamoDbClient.batchWriteItem(b ->  
b.requestItems(productCatalogItems));
```

```
    logger.info("Unprocessed items: " + response.unprocessedItems().size());
}
```

## Confronto: Scrittura in batch utilizzando il DynamoDB Enhanced Client

```
public static void batchWriteEnhanced(DynamoDbTable<ProductCatalog> productCatalog)
{
    ProductCatalog prod = ProductCatalog.builder()
        .id(1)
        .isbn("1-565-85698")
        .authors(new HashSet<>(Arrays.asList("a", "b")))
        .price(BigDecimal.valueOf(52.13))
        .title("Title 1")
        .build();
    ProductCatalog prod2 = ProductCatalog.builder()
        .id(2)
        .isbn("1-208-98073")
        .authors(new HashSet<>(Arrays.asList("a", "b", "c")))
        .price(BigDecimal.valueOf(21.99))
        .title("Title 2")
        .build();
    ProductCatalog prod3 = ProductCatalog.builder()
        .id(3)
        .isbn("7-236-98618")
        .authors(new HashSet<>(Arrays.asList("g", "k", "c")))
        .price(BigDecimal.valueOf(42.00))
        .title("Title 3")
        .build();

    BatchWriteResult batchWriteResult = DependencyFactory.enhancedClient()
        .batchWriteItem(b -> b.writeBatches(
            WriteBatch.builder(ProductCatalog.class)
                .mappedTableResource(productCatalog)
                .addPutItem(prod).addPutItem(prod2).addPutItem(prod3)
                .build()
        ));
    logger.info("Unprocessed items: " +
batchWriteResult.unprocessedPutItemsForTable(productCatalog).size());
}
```

## Lavora con classi di dati immutabili

La funzionalità di mappatura dell'API DynamoDB Enhanced Client funziona con classi di dati immutabili. Una classe immutabile ha solo getter e richiede una classe builder che l'SDK utilizza per creare istanze della classe. Invece di utilizzare l'`@DynamoDbBean` annotazione come mostrato nella [classe Customer](#), le classi immutabili utilizzano l'`@DynamoDbImmutable` annotazione, che accetta un parametro che indica la classe builder da utilizzare.

La classe seguente è una versione immutabile di `Customer`

```
package org.example.tests.model.immutable;

import software.amazon.awssdk.enhanced.dynamodb.mapper.annotations.DynamoDbImmutable;
import
    software.amazon.awssdk.enhanced.dynamodb.mapper.annotations.DynamoDbPartitionKey;
import
    software.amazon.awssdk.enhanced.dynamodb.mapper.annotations.DynamoDbSecondaryPartitionKey;
import
    software.amazon.awssdk.enhanced.dynamodb.mapper.annotations.DynamoDbSecondarySortKey;
import software.amazon.awssdk.enhanced.dynamodb.mapper.annotations.DynamoDbSortKey;

import java.time.Instant;

@DynamoDbImmutable(builder = CustomerImmutable.Builder.class)
public class CustomerImmutable {
    private final String id;
    private final String name;
    private final String email;
    private final Instant regDate;

    private CustomerImmutable(Builder b) {
        this.id = b.id;
        this.email = b.email;
        this.name = b.name;
        this.regDate = b.regDate;
    }

    // This method will be automatically discovered and used by the TableSchema.
    public static Builder builder() { return new Builder(); }

    @DynamoDbPartitionKey
    public String id() { return this.id; }
```



```

@DynamoDbSortKey
public String email() { return this.email; }

@DynamoDbSecondaryPartitionKey(indexNames = "customers_by_name")
public String name() { return this.name; }

@DynamoDbSecondarySortKey(indexNames = {"customers_by_date", "customers_by_name"})
public Instant regDate() { return this.regDate; }

public static final class Builder {
    private String id;
    private String email;
    private String name;
    private Instant regDate;

    // The private Builder constructor is visible to the enclosing
    CustomerImmutable class.
    private Builder() {}

    public Builder id(String id) { this.id = id; return this; }
    public Builder email(String email) { this.email = email; return this; }
    public Builder name(String name) { this.name = name; return this; }
    public Builder regDate(Instant regDate) { this.regDate = regDate; return
this; }

    // This method will be automatically discovered and used by the TableSchema.
    public CustomerImmutable build() { return new CustomerImmutable(this); }
}
}

```

È necessario soddisfare i seguenti requisiti quando si annota una classe di dati con.

### @DynamoDbImmutable

1. Ogni metodo che non sostituisce `Object.class` e con cui non è stato annotato `@DynamoDbIgnore` deve essere un getter per un attributo della tabella DynamoDB.
2. Ogni getter deve avere un setter corrispondente con distinzione tra maiuscole e minuscole nella classe builder.
3. Deve essere soddisfatta solo una delle seguenti condizioni di costruzione.
  - La classe builder deve avere un costruttore pubblico predefinito.

- La classe di dati deve avere un metodo statico pubblico denominato `builder()` che non accetta parametri e restituisce un'istanza della classe builder. Questa opzione è mostrata nella classe `immutableCustomer`.
4. La classe builder deve avere un metodo pubblico denominato `build()` che non accetta parametri e restituisce un'istanza della classe immutabile.

Per crearne uno `TableSchema` per la tua classe immutabile, usa il `fromImmutableClass()` metodo on `TableSchema` come mostrato nel frammento seguente.

```
static final TableSchema<CustomerImmutable> customerImmutableTableSchema =
    TableSchema.fromImmutableClass(CustomerImmutable.class);
```

Proprio come è possibile creare una tabella DynamoDB da una classe mutabile, è possibile crearne una da una classe immutabile con una chiamata singola a `of`, come illustrato nel seguente esempio `DynamoDbTable` di `createTable()` snippet.

```
static void createTableFromImmutable(DynamoDbEnhancedClient enhancedClient, String
    tableName, DynamoDbWaiter waiter){
    // First, create an in-memory representation of the table using the 'table()'
    method of the DynamoDb Enhanced Client.
    // 'table()' accepts a name for the table and a TableSchema instance that you
    created previously.
    DynamoDbTable<CustomerImmutable> customerDynamoDbTable = enhancedClient
        .table(tableName, TableSchema.fromImmutableClass(CustomerImmutable.class));

    // Second, call the 'createTable()' method on the DynamoDbTable instance.
    customerDynamoDbTable.createTable();
    waiter.waitUntilTableExists(b -> b.tableName(tableName));
}
```

Usa librerie di terze parti, come Lombok

Le librerie di terze parti, come [Project Lombok](#), aiutano a generare codice standard associato a oggetti immutabili. L'API DynamoDB Enhanced Client funziona con queste librerie purché le classi di dati seguano le convenzioni descritte in questa sezione.

L'esempio seguente mostra la classe immutabile `CustomerImmutable` con annotazioni Lombok. Nota come la `onMethod` funzionalità di Lombok copia le annotazioni DynamoDB basate su attributi, ad esempio, sul codice generato. `@DynamoDbPartitionKey`

```
@Value
@Builder
@DynamoDbImmutable(builder = Customer.CustomerBuilder.class)
public class Customer {
    @Getter(onMethod_=@DynamoDbPartitionKey)
    private String id;

    @Getter(onMethod_=@DynamoDbSortKey)
    private String email;

    @Getter(onMethod_=@DynamoDbSecondaryPartitionKey(indexNames = "customers_by_name"))
    private String name;

    @Getter(onMethod_=@DynamoDbSecondarySortKey(indexNames = {"customers_by_date",
"customers_by_name"}))
    private Instant createdAt;
}
```

Usa espressioni e condizioni

[Le espressioni nell'API DynamoDB Enhanced Client sono rappresentazioni Java delle espressioni DynamoDB.](#)

L'API DynamoDB Enhanced Client utilizza tre tipi di espressioni:

### [Expression](#)

La Expression classe viene utilizzata quando si definiscono condizioni e filtri.

### [QueryConditional](#)

Questo tipo di espressione rappresenta [le condizioni chiave per le](#) operazioni di interrogazione.

### [UpdateExpression](#)

Questa classe consente di scrivere espressioni di [aggiornamento di DynamoDB](#) ed è attualmente utilizzata nel framework di estensione quando si aggiorna un elemento.

## Anatomia delle espressioni

Un'espressione è composta dai seguenti elementi:

- Un'espressione stringa (obbligatoria). La stringa contiene un'espressione logica DynamoDB con nomi segnaposto per i nomi e i valori degli attributi.

- Una mappa dei valori delle espressioni (in genere richiesta).
- Una mappa dei nomi delle espressioni (opzionale).

Utilizzate un builder per generare un Expression oggetto che assuma la seguente forma generale.

```
Expression expression = Expression.builder()
    .expression(<String>)
    .expressionNames(<Map>)
    .expressionValues(<Map>)
    .build()
```

ExpressionDi solito richiede una mappa di valori di espressione. La mappa fornisce i valori per i segnaposto nell'espressione stringa. La chiave della mappa è costituita dal nome del segnaposto preceduto da due punti (:) e il valore della mappa è un'istanza di [AttributeValue](#). La [AttributeValues](#) classe dispone di metodi pratici per generare un'AttributeValueistanza da un valore letterale. In alternativa, è possibile utilizzare il `AttributeValue.Builder` per generare un'AttributeValueistanza.

Il seguente frammento mostra una mappa con due voci dopo la riga di commento 2. La stringa passata al `expression()` metodo, mostrata dopo la riga di commento 1, contiene i segnaposto che DynamoDB risolve prima di eseguire l'operazione. Questo frammento non contiene una mappa di nomi di espressioni, perché `price` è un nome di attributo consentito.

```
public static void scanAsync(DynamoDbAsyncTable productCatalog) {
    ScanEnhancedRequest request = ScanEnhancedRequest.builder()
        .consistentRead(true)
        .attributesToProject("id", "title", "authors", "price")
        .filterExpression(Expression.builder()
            // 1. :min_value and :max_value are placeholders for the values
            // provided by the map
            .expression("price >= :min_value AND price <= :max_value")
            // 2. Two values are needed for the expression and each is
            // supplied as a map entry.
            .expressionValues(
                Map.of( ":min_value", numberValue(8.00),
                    ":max_value", numberValue(400_000.00)))
            .build())
        .build();
}
```

Se il nome di un attributo nella tabella DynamoDB è una parola riservata, inizia con un numero o contiene uno spazio, è necessaria una mappa dei nomi delle espressioni per. `Expression`

Ad esempio, se il nome dell'attributo era `1price` al posto dell'`price` esempio di codice precedente, l'esempio dovrebbe essere modificato come mostrato nell'esempio seguente.

```
ScanEnhancedRequest request = ScanEnhancedRequest.builder()
    .filterExpression(Expression.builder()
        .expression("#price >= :min_value AND #price <= :max_value")
        .expressionNames( Map.of("#price", "1price") )
        .expressionValues(
            Map.of(":min_value", numberValue(8.00),
                ":max_value", numberValue(400_000.00)))
        .build())
    .build();
```

Un segnaposto per il nome di un'espressione inizia con il simbolo del cancelletto (`#`). Una voce per la mappa dei nomi delle espressioni utilizza il segnaposto come chiave e il nome dell'attributo come valore. La mappa viene aggiunta al generatore di espressioni con il metodo `expressionNames()`. DynamoDB risolve il nome dell'attributo prima di eseguire l'operazione.

I valori di espressione non sono necessari se una funzione viene utilizzata nell'espressione stringa. Un esempio di funzione di espressione è `attribute_exists(<attribute_name>)`.

L'esempio seguente crea un file `Expression` che utilizza una funzione [DynamoDB](#). La stringa di espressione in questo esempio non utilizza segnaposto. Questa espressione può essere utilizzata in un'operazione `putItem` per verificare se nel database esiste già un elemento con un valore dell'attributo uguale all'attributo dell'oggetto dati.

```
Expression exp = Expression.builder().expression("attribute_not_exists
(movie)").build();
```

La [DynamoDB Developer Guide](#) contiene informazioni complete sulle espressioni di [basso livello utilizzate](#) con DynamoDB.

## Espressioni condizionali e condizionali

Quando si utilizzano i metodi `deleteItem()`, `putItem()`, `updateItem()`, e anche quando si utilizzano operazioni di transazione e batch, si utilizzano [Expression](#) oggetti per specificare le

condizioni che DynamoDB deve soddisfare per procedere con l'operazione. Queste espressioni sono denominate espressioni condizionali. Per un esempio, vedi l'espressione di condizione utilizzata nel `addDeleteItem()` metodo (dopo la riga di commento 1) dell'[esempio di transazione](#) mostrato in questa guida.

Quando si utilizzano i `query()` metodi, una condizione viene espressa come a [QueryConditional](#). La `QueryConditional` classe dispone di diversi metodi statici di convenienza che aiutano a scrivere i criteri che determinano quali elementi leggere da DynamoDB.

Per alcuni esempi di `QueryConditionals`, consultate il primo esempio di codice della [the section called "Queryesempi di metodi"](#) sezione di questa guida.

## Espressioni filtro

Le espressioni di filtro vengono utilizzate nelle operazioni di scansione e interrogazione per filtrare gli elementi restituiti.

Un'espressione di filtro viene applicata dopo che tutti i dati sono stati letti dal database, quindi il costo di lettura è lo stesso che se non ci fosse alcun filtro. [La Amazon DynamoDB Developer Guide contiene ulteriori informazioni sull'uso delle espressioni di filtro per le operazioni di interrogazione e scansione.](#)

L'esempio seguente mostra un'espressione di filtro aggiunta a una richiesta di scansione. I criteri limitano gli articoli restituiti agli articoli con un prezzo compreso tra 8,00 e 80,00€ inclusi.

```
Map<String, AttributeValue> expressionValues = Map.of(
    ":min_value", numberValue(8.00),
    ":max_value", numberValue(80.00));

ScanEnhancedRequest request = ScanEnhancedRequest.builder()
    .consistentRead(true)
    // 1. the 'attributesToProject()' method allows you to specify which
values you want returned.
    .attributesToProject("id", "title", "authors", "price")
    // 2. Filter expression limits the items returned that match the
provided criteria.
    .filterExpression(Expression.builder()
        .expression("price >= :min_value AND price <= :max_value")
        .expressionValues(expressionValues)
        .build())
    .build();
```

## Espressioni di aggiornamento

Il metodo di DynamoDB Enhanced Client `updateItem()` fornisce un modo standard per aggiornare gli elementi in DynamoDB. [Tuttavia, quando hai bisogno di maggiori funzionalità, UpdateExpressions fornisce una rappresentazione sicura dei tipi della sintassi delle espressioni di aggiornamento di DynamoDB.](#) Ad esempio, è possibile utilizzare UpdateExpressions per aumentare i valori senza prima leggere elementi da DynamoDB o aggiungere singoli membri a un elenco. Le espressioni di aggiornamento sono attualmente disponibili nelle estensioni personalizzate per il `updateItem()` metodo.

Per un esempio che utilizza le espressioni di aggiornamento, consultate l'[esempio di estensione personalizzata](#) in questa guida.

Ulteriori informazioni sulle espressioni di aggiornamento sono disponibili nella [Amazon DynamoDB Developer Guide](#).

## Lavora con risultati impaginati: scansioni e interrogazioni

I *scan batch* metodi *query* e dell'API DynamoDB Enhanced Client restituiscono risposte con una o più pagine. Una pagina contiene uno o più elementi. Il codice può elaborare la risposta per pagina o elaborare singoli elementi.

Una risposta impaginata restituita dal client sincrono restituisce un [PageIterable](#) oggetto, mentre una risposta restituita dal `DynamoDbEnhancedClient` client asincrono restituisce un oggetto `DynamoDbEnhancedAsyncClient` [PagePublisher](#)

Questa sezione esamina l'elaborazione dei risultati impaginati e fornisce esempi che utilizzano le API di scansione e interrogazione.

## Esegui la scansione di una tabella

Il [scan](#) metodo dell'SDK corrisponde all'operazione [DynamoDB con lo stesso nome](#). L'API DynamoDB Enhanced Client offre le stesse opzioni, ma utilizza un modello a oggetti familiare e gestisce l'impaginazione al posto tuo.

Innanzitutto, esploriamo l'`PageIterable` interfaccia esaminando il `scan` metodo della classe di mappatura sincrona, [DynamoDbTable](#)

## Usa l'API sincrona

L'esempio seguente mostra il `scan` metodo che utilizza un [espressione](#) per filtrare gli elementi restituiti. [ProductCatalog](#) È l'oggetto modello che è stato mostrato in precedenza.

L'espressione di filtro mostrata dopo la riga di commento 1 limita gli `ProductCatalog` articoli che vengono restituiti a quelli con un valore di prezzo compreso tra 8,00 e 80,00 inclusi.

Questo esempio esclude anche `isbn` i valori utilizzando il `attributesToProject` metodo mostrato dopo la riga di commento 2.

Nella riga di commento 3, l'`PageIterable` oggetto, `pagedResult`, viene restituito dal `scan` metodo. Il `stream` metodo di `PageIterable` restituisce un [java.util.Stream](#) oggetto, che è possibile utilizzare per elaborare le pagine. In questo esempio, il numero di pagine viene contato e registrato.

A partire dalla riga di commento 4, l'esempio mostra due varianti di accesso agli `ProductCatalog` elementi. La versione successiva alla riga di commento 2a scorre attraverso ogni pagina e ordina e registra gli elementi su ogni pagina. La versione successiva alla riga di commento 2b salta l'iterazione della pagina e accede direttamente agli elementi.

L'`PageIterable` interfaccia offre diversi modi per elaborare i risultati grazie alle sue due interfacce principali: e. [java.lang.IterableSdkIterable](#) `Iterable` porta i `splitter` e `iterator` metodi, `iterator` e `SdkIterable` porta il `stream` metodo.

```
public static void scanSync(DynamoDbTable<ProductCatalog> productCatalog) {

    Map<String, AttributeValue> expressionValues = Map.of(
        ":min_value", numberValue(8.00),
        ":max_value", numberValue(80.00));

    ScanEnhancedRequest request = ScanEnhancedRequest.builder()
        .consistentRead(true)
        // 1. the 'attributesToProject()' method allows you to specify which
values you want returned.
        .attributesToProject("id", "title", "authors", "price")
        // 2. Filter expression limits the items returned that match the
provided criteria.
        .filterExpression(Expression.builder()
            .expression("price >= :min_value AND price <= :max_value")
            .expressionValues(expressionValues)
            .build())
        .build();

    // 3. A PageIterable object is returned by the scan method.
    PageIterable<ProductCatalog> pagedResults = productCatalog.scan(request);
    logger.info("page count: {}", pagedResults.stream().count());
}
```



```

// 4. Log the returned ProductCatalog items using two variations.
// 4a. This version sorts and logs the items of each page.
pagedResults.stream().forEach(p -> p.items().stream()
    .sorted(Comparator.comparing(ProductCatalog::price))
    .forEach(
        item -> logger.info(item.toString())
    ));
// 4b. This version sorts and logs all items for all pages.
pagedResults.items().stream()
    .sorted(Comparator.comparing(ProductCatalog::price))
    .forEach(
        item -> logger.info(item.toString())
    );
}

```

## Usa l'API asincrona

Il `scan` metodo asincrono restituisce i risultati come oggetto. `PagePublisher` L'`PagePublisher` interfaccia dispone di due `subscribe` metodi che è possibile utilizzare per elaborare le pagine di risposta. Un `subscribe` metodo proviene dall'interfaccia `org.reactivestreams.Publisher` principale. Per elaborare le pagine utilizzando questa prima opzione, passate un'[Subscriber](#) istanza al `subscribe` metodo. Il primo esempio che segue mostra l'uso del `subscribe` metodo.

Il secondo `subscribe` metodo proviene dall'[SdkPublisher](#) interfaccia. Questa versione di `subscribe` accetta a [Consumer](#) anziché a `Subscriber`. Questa variazione di `subscribe` metodo è illustrata nel secondo esempio che segue.

L'esempio seguente mostra la versione asincrona del `scan` metodo che utilizza la stessa espressione di filtro mostrata nell'esempio precedente.

Dopo la riga di commento 3, `DynamoDbAsyncTable.scan` restituisce un oggetto. `PagePublisher` Nella riga successiva, il codice crea un'istanza dell'`org.reactivestreams.Subscriber` interfaccia `ProductCatalogSubscriber`, che sottoscrive la riga 4 `PagePublisher` dopo il commento.

L'`Subscriber` oggetto raccoglie gli `ProductCatalog` elementi da ogni pagina del `onNext` metodo dopo la riga di commento 8 nell'esempio di `ProductCatalogSubscriber` classe. Gli elementi sono memorizzati nella `List` variabile privata e vi si accede nel codice di chiamata con il `ProductCatalogSubscriber.getSubscribedItems()` metodo. Viene chiamato dopo la riga di commento 5.

Dopo aver recuperato l'elenco, il codice ordina tutti gli `ProductCatalog` articoli in base al prezzo e registra ogni articolo.

La `ProductCatalogSubscriber` classe [CountDownLatch](#) in the blocca il thread chiamante fino a quando tutti gli elementi non sono stati aggiunti all'elenco prima di continuare dopo la riga di commento 5.

```
public static void scanAsync(DynamoDbAsyncTable productCatalog) {
    ScanEnhancedRequest request = ScanEnhancedRequest.builder()
        .consistentRead(true)
        .attributesToProject("id", "title", "authors", "price")
        .filterExpression(Expression.builder()
            // 1. :min_value and :max_value are placeholders for the values
provided by the map
            .expression("price >= :min_value AND price <= :max_value")
            // 2. Two values are needed for the expression and each is
supplied as a map entry.
            .expressionValues(
                Map.of( ":min_value", numberValue(8.00),
                    ":max_value", numberValue(400_000.00)))
            .build())
        .build();

    // 3. A PagePublisher object is returned by the scan method.
    PagePublisher<ProductCatalog> pagePublisher = productCatalog.scan(request);
    ProductCatalogSubscriber subscriber = new ProductCatalogSubscriber();
    // 4. Subscribe the ProductCatalogSubscriber to the PagePublisher.
    pagePublisher.subscribe(subscriber);
    // 5. Retrieve all collected ProductCatalog items accumulated by the
subscriber.
    subscriber.getSubscribedItems().stream()
        .sorted(Comparator.comparing(ProductCatalog::price))
        .forEach(item ->
            logger.info(item.toString()));
    // 6. Use a Consumer to work through each page.
    pagePublisher.subscribe(page -> page
        .items().stream()
        .sorted(Comparator.comparing(ProductCatalog::price))
        .forEach(item ->
            logger.info(item.toString())))
        .join(); // If needed, blocks the subscribe() method thread until it is
finished processing.
    // 7. Use a Consumer to work through each ProductCatalog item.
```

```

    pagePublisher.items()
        .subscribe(product -> logger.info(product.toString()))
        .exceptionally(failure -> {
            logger.error("ERROR - ", failure);
            return null;
        })
        .join(); // If needed, blocks the subscribe() method thread until it is
finished processing.
    }

```

```

private static class ProductCatalogSubscriber implements
Subscriber<Page<ProductCatalog>> {
    private CountdownLatch latch = new CountdownLatch(1);
    private Subscription subscription;
    private List<ProductCatalog> itemsFromAllPages = new ArrayList<>();

    @Override
    public void onSubscribe(Subscription sub) {
        subscription = sub;
        subscription.request(1L);
        try {
            latch.await(); // Called by main thread blocking it until latch is
released.
        } catch (InterruptedException e) {
            throw new RuntimeException(e);
        }
    }

    @Override
    public void onNext(Page<ProductCatalog> productCatalogPage) {
        // 8. Collect all the ProductCatalog instances in the page, then ask the
publisher for one more page.
        itemsFromAllPages.addAll(productCatalogPage.items());
        subscription.request(1L);
    }

    @Override
    public void onError(Throwable throwable) {
    }

    @Override
    public void onComplete() {
        latch.countDown(); // Call by subscription thread; latch releases.
    }
}

```

```

    }

    List<ProductCatalog> getSubscribedItems() {
        return this.itemsFromAllPages;
    }
}

```

Il seguente esempio di frammento utilizza la versione del `PagePublisher.subscribe` metodo che accetta una riga `Consumer` successiva al commento 6. Il parametro Java lambda utilizza le pagine, che elaborano ulteriormente ogni elemento. In questo esempio, ogni pagina viene elaborata e gli elementi di ogni pagina vengono ordinati e quindi registrati.

```

// 6. Use a Consumer to work through each page.
pagePublisher.subscribe(page -> page
    .items().stream()
    .sorted(Comparator.comparing(ProductCatalog::price))
    .forEach(item ->
        logger.info(item.toString())))
    .join(); // If needed, blocks the subscribe() method thread until it is
finished processing.

```

Il `items` metodo di `PagePublisher` decodifica le istanze del modello in modo che il codice possa elaborare direttamente gli elementi. Questo approccio è illustrato nel frammento seguente.

```

// 7. Use a Consumer to work through each ProductCatalog item.
pagePublisher.items()
    .subscribe(product -> logger.info(product.toString()))
    .exceptionally(failure -> {
        logger.error("ERROR - ", failure);
        return null;
    })
    .join(); // If needed, blocks the subscribe() method thread until it is
finished processing.

```

## Esecuzione di una query su una tabella

Il [`query\(\)`](#) metodo della `DynamoDbTable` classe trova gli elementi in base ai valori della chiave primaria. L'`@DynamoDbPartitionKey` annotazione e l'`@DynamoDbSortKey` annotazione opzionale vengono utilizzate per definire la chiave primaria sulla classe di dati.

Il `query()` metodo richiede un valore della chiave di partizione che trovi gli elementi che corrispondono al valore fornito. Se la tabella definisce anche una chiave di ordinamento, è possibile aggiungerne un valore alla query come condizione di confronto aggiuntiva per ottimizzare i risultati.

Fatta eccezione per l'elaborazione dei risultati, le versioni sincrona e asincrona di `query()` funzionano allo stesso modo. Come con `scanAPI`, `queryAPI` restituisce un `PageIterable` per una chiamata sincrona e uno per una chiamata asincrona. `PagePublisher` Abbiamo discusso l'uso di `PageIterable` e in `PagePublisher` precedenza nella sezione di scansione.

## Query esempi di metodi

L'esempio di codice del `query()` metodo che segue utilizza la `MovieActor` classe. La classe di dati definisce una chiave primaria composta composta dall'`movie` attributo per la chiave di partizione e dall'`actor` attributo per la chiave di ordinamento.

La classe segnala inoltre che utilizza un indice secondario globale denominato `acting_award_year`. La chiave primaria composta dell'indice è composta dall'`actingaward` attributo per la chiave di partizione e dall'attributo `actingyear` per la chiave di ordinamento. Più avanti in questo argomento, quando mostreremo come creare e utilizzare gli indici, faremo riferimento all'indice. `acting_award_year`

## Classe `MovieActor`

```
package org.example.tests.model;

import software.amazon.awssdk.enhanced.dynamodb.mapper.annotations.DynamoDbAttribute;
import software.amazon.awssdk.enhanced.dynamodb.mapper.annotations.DynamoDbBean;
import
    software.amazon.awssdk.enhanced.dynamodb.mapper.annotations.DynamoDbPartitionKey;
import
    software.amazon.awssdk.enhanced.dynamodb.mapper.annotations.DynamoDbSecondaryPartitionKey;
import
    software.amazon.awssdk.enhanced.dynamodb.mapper.annotations.DynamoDbSecondarySortKey;
import software.amazon.awssdk.enhanced.dynamodb.mapper.annotations.DynamoDbSortKey;

import java.util.Objects;

@DynamoDbBean
public class MovieActor implements Comparable<MovieActor> {

    private String movieName;
    private String actorName;
```

```
private String actingAward;
private Integer actingYear;
private String actingSchoolName;

@DynamoDbPartitionKey
@DynamoDbAttribute("movie")
public String getMovieName() {
    return movieName;
}

public void setMovieName(String movieName) {
    this.movieName = movieName;
}

@DynamoDbSortKey
@DynamoDbAttribute("actor")
public String getActorName() {
    return actorName;
}

public void setActorName(String actorName) {
    this.actorName = actorName;
}

@DynamoDbSecondaryPartitionKey(indexNames = "acting_award_year")
@DynamoDbAttribute("actingaward")
public String getActingAward() {
    return actingAward;
}

public void setActingAward(String actingAward) {
    this.actingAward = actingAward;
}

@DynamoDbSecondarySortKey(indexNames = {"acting_award_year", "movie_year"})
@DynamoDbAttribute("actingyear")
public Integer getActingYear() {
    return actingYear;
}

public void setActingYear(Integer actingYear) {
    this.actingYear = actingYear;
}
```

```
@DynamoDbAttribute("actingschoolname")
public String getActingSchoolName() {
    return actingSchoolName;
}

public void setActingSchoolName(String actingSchoolName) {
    this.actingSchoolName = actingSchoolName;
}

@Override
public String toString() {
    final StringBuffer sb = new StringBuffer("MovieActor{");
    sb.append("movieName=").append(movieName).append('\ ');
    sb.append(", actorName=").append(actorName).append('\ ');
    sb.append(", actingAward=").append(actingAward).append('\ ');
    sb.append(", actingYear=").append(actingYear);
    sb.append(", actingSchoolName=").append(actingSchoolName).append('\ ');
    sb.append('}');
    return sb.toString();
}

@Override
public boolean equals(Object o) {
    if (this == o) return true;
    if (o == null || getClass() != o.getClass()) return false;
    MovieActor that = (MovieActor) o;
    return Objects.equals(movieName, that.movieName) && Objects.equals(actorName,
that.actorName) && Objects.equals(actingAward, that.actingAward) &&
Objects.equals(actingYear, that.actingYear) && Objects.equals(actingSchoolName,
that.actingSchoolName);
}

@Override
public int hashCode() {
    return Objects.hash(movieName, actorName, actingAward, actingYear,
actingSchoolName);
}

@Override
public int compareTo(MovieActor o) {
    if (this.movieName.compareTo(o.movieName) != 0){
        return this.movieName.compareTo(o.movieName);
    } else {
        return this.actorName.compareTo(o.actorName);
    }
}
```

```
    }  
  }  
}
```

Gli esempi di codice che seguono eseguono una query sui seguenti elementi.

### Elementi nella **MovieActor** tabella

```
MovieActor{movieName='movie01', actorName='actor0', actingAward='actingaward0',  
  actingYear=2001, actingSchoolName='null'}  
MovieActor{movieName='movie01', actorName='actor1', actingAward='actingaward1',  
  actingYear=2001, actingSchoolName='actingschool1'}  
MovieActor{movieName='movie01', actorName='actor2', actingAward='actingaward2',  
  actingYear=2001, actingSchoolName='actingschool2'}  
MovieActor{movieName='movie01', actorName='actor3', actingAward='actingaward3',  
  actingYear=2001, actingSchoolName='null'}  
MovieActor{movieName='movie01', actorName='actor4', actingAward='actingaward4',  
  actingYear=2001, actingSchoolName='actingschool4'}  
MovieActor{movieName='movie02', actorName='actor0', actingAward='actingaward0',  
  actingYear=2002, actingSchoolName='null'}  
MovieActor{movieName='movie02', actorName='actor1', actingAward='actingaward1',  
  actingYear=2002, actingSchoolName='actingschool1'}  
MovieActor{movieName='movie02', actorName='actor2', actingAward='actingaward2',  
  actingYear=2002, actingSchoolName='actingschool2'}  
MovieActor{movieName='movie02', actorName='actor3', actingAward='actingaward3',  
  actingYear=2002, actingSchoolName='null'}  
MovieActor{movieName='movie02', actorName='actor4', actingAward='actingaward4',  
  actingYear=2002, actingSchoolName='actingschool4'}  
MovieActor{movieName='movie03', actorName='actor0', actingAward='actingaward0',  
  actingYear=2003, actingSchoolName='null'}  
MovieActor{movieName='movie03', actorName='actor1', actingAward='actingaward1',  
  actingYear=2003, actingSchoolName='actingschool1'}  
MovieActor{movieName='movie03', actorName='actor2', actingAward='actingaward2',  
  actingYear=2003, actingSchoolName='actingschool2'}  
MovieActor{movieName='movie03', actorName='actor3', actingAward='actingaward3',  
  actingYear=2003, actingSchoolName='null'}  
MovieActor{movieName='movie03', actorName='actor4', actingAward='actingaward4',  
  actingYear=2003, actingSchoolName='actingschool4'}
```

Il codice seguente definisce due [QueryConditional](#)istanze. [QueryConditional](#)sfunzionano con i valori chiave, la sola chiave di partizione o in combinazione con la chiave di ordinamento, e corrispondono alle [espressioni condizionali](#) chiave dell'API del servizio DynamoDB. Dopo la riga di



commento 1, l'esempio definisce l'`keyEqual` istanza che corrisponde agli elementi con un valore di partizione di **movie01**

Questo esempio definisce anche un'espressione di filtro che filtra qualsiasi elemento che non è attivo dopo la **actingschoolname** di commento 2.

Dopo la riga di commento 3, l'esempio mostra l'[QueryEnhancedRequest](#) istanza che il codice passa al `DynamoDbTable.query()` metodo. Questo oggetto combina la condizione chiave e il filtro utilizzati dall'SDK per generare la richiesta al servizio DynamoDB.

```
public static void query(DynamoDbTable movieActorTable) {

    // 1. Define a QueryConditional instance to return items matching a partition
    value.
    QueryConditional keyEqual = QueryConditional.keyEqualTo(b ->
    b.partitionValue("movie01"));
    // 1a. Define a QueryConditional that adds a sort key criteria to the partition
    value criteria.
    QueryConditional sortGreaterThanOrEqualTo =
    QueryConditional.sortGreaterThanOrEqualTo(b ->
    b.partitionValue("movie01").sortValue("actor2"));
    // 2. Define a filter expression that filters out items whose attribute value
    is null.
    final Expression filterOutNoActingschoolname =
    Expression.builder().expression("attribute_exists(actingschoolname)").build();

    // 3. Build the query request.
    QueryEnhancedRequest tableQuery = QueryEnhancedRequest.builder()
        .queryConditional(keyEqual)
        .filterExpression(filterOutNoActingschoolname)
        .build();
    // 4. Perform the query.
    PageIterable<MovieActor> pagedResults = movieActorTable.query(tableQuery);
    logger.info("page count: {}", pagedResults.stream().count()); // Log number of
    pages.

    pagedResults.items().stream()
        .sorted()
        .forEach(
            item -> logger.info(item.toString()) // Log the sorted list of
    items.
        );
}
```

Di seguito è riportato l'output dell'esecuzione del metodo. L'output mostra gli elementi con un `movieName` valore di `movie01` e non visualizza alcun elemento `actingSchoolName` uguale a `null`

```
2023-03-05 13:11:05 [main] INFO org.example.tests.QueryDemo:46 - page count: 1
2023-03-05 13:11:05 [main] INFO org.example.tests.QueryDemo:51 -
  MovieActor{movieName='movie01', actorName='actor1', actingAward='actingaward1',
  actingYear=2001, actingSchoolName='actingschool1'}
2023-03-05 13:11:05 [main] INFO org.example.tests.QueryDemo:51 -
  MovieActor{movieName='movie01', actorName='actor2', actingAward='actingaward2',
  actingYear=2001, actingSchoolName='actingschool2'}
2023-03-05 13:11:05 [main] INFO org.example.tests.QueryDemo:51 -
  MovieActor{movieName='movie01', actorName='actor4', actingAward='actingaward4',
  actingYear=2001, actingSchoolName='actingschool4'}
```

Nella seguente variante della richiesta di interrogazione mostrata in precedenza dopo la riga di commento 3, il codice sostituisce `sortGreaterThanOrEqualTo` `QueryConditional` quella definita dopo la riga di commento 1a. `keyEqual` `QueryConditional` Il codice seguente rimuove anche l'espressione del filtro.

```
QueryEnhancedRequest tableQuery = QueryEnhancedRequest.builder()
    .queryConditional(sortGreaterThanOrEqualTo)
```

Poiché questa tabella ha una chiave primaria composta, tutte le `QueryConditional` istanze richiedono un valore di chiave di partizione. `QueryConditional` metodi che iniziano con `sort...` indicano che è necessaria una chiave di ordinamento. I risultati non vengono ordinati.

L'output seguente mostra i risultati dell'interrogazione. La query restituisce gli elementi con un `movieName` valore uguale a `movie01` e solo gli elementi che hanno un `actorName` valore maggiore o uguale a `actor2`. Poiché il filtro è stato rimosso, la query restituisce elementi che non hanno alcun valore per l'attributo. `actingSchoolName`

```
2023-03-05 13:15:00 [main] INFO org.example.tests.QueryDemo:46 - page count: 1
2023-03-05 13:15:00 [main] INFO org.example.tests.QueryDemo:51 -
  MovieActor{movieName='movie01', actorName='actor2', actingAward='actingaward2',
  actingYear=2001, actingSchoolName='actingschool2'}
2023-03-05 13:15:00 [main] INFO org.example.tests.QueryDemo:51 -
  MovieActor{movieName='movie01', actorName='actor3', actingAward='actingaward3',
  actingYear=2001, actingSchoolName='null'}
```

```
2023-03-05 13:15:00 [main] INFO org.example.tests.QueryDemo:51 -
MovieActor{movieName='movie01', actorName='actor4', actingAward='actingaward4',
actingYear=2001, actingSchoolName='actingschool4'}
```

Eseguire operazioni in batch

[L'API DynamoDB Enhanced Client offre due metodi batch `batchGetItem\(\)` e `batchWriteItem\(\)`.](#)

### `batchGetItem()` Esempio

Con [`DynamoDbTable.batchGetItem\(\)`](#) questo metodo, è possibile recuperare fino a 100 singoli elementi su più tabelle in un'unica richiesta complessiva. L'esempio seguente utilizza le classi di [`MovieActor`](#) e [`Customer`](#) mostrate in precedenza.

Nell'esempio dopo le righe 1 e 2, create [`ReadBatch`](#) oggetti che successivamente aggiungete come parametri al `batchGetItem()` metodo dopo la riga di commento 3. Il codice dopo la riga di commento 1 crea il batch da leggere dalla `Customer` tabella. Il codice dopo la riga di commento 1a mostra l'uso di un [`GetItemEnhancedRequest`](#) generatore che utilizza i valori della chiave primaria per specificare l'elemento da leggere. A differenza di specificare i valori chiave per richiedere un elemento, è possibile utilizzare una classe di dati per richiedere un elemento, come mostrato dopo la riga di commento 1b. L'SDK estrae i valori chiave dietro le quinte prima di inviare la richiesta.

[Quando si specifica l'elemento utilizzando l'approccio basato su chiavi, come mostrato nelle due istruzioni successive a 2a, è anche possibile specificare che DynamoDB deve eseguire una lettura fortemente coerente.](#) Quando viene utilizzato, il `consistentRead()` metodo deve essere utilizzato su tutti gli elementi richiesti per la stessa tabella.

Per recuperare gli elementi trovati da DynamoDB, utilizzate [`resultsForTable\(\)`](#) il metodo mostrato dopo la riga di commento 4. Chiamate il metodo per ogni tabella che è stata letta nella richiesta. `resultsForTable()` restituisce un elenco di elementi trovati che è possibile elaborare utilizzando qualsiasi `java.util.List` metodo. Questo esempio registra ogni elemento.

Per scoprire elementi che DynamoDB non ha elaborato, utilizzate l'approccio dopo la riga di commento 5. La `BatchGetResultPage` classe utilizza il [`unprocessedKeysForTable\(\)`](#) metodo che consente di accedere a ogni chiave non elaborata. Il [riferimento all'BatchGetItem API](#) contiene ulteriori informazioni sulle situazioni che generano elementi non elaborati.

```
public static void batchGetItemExample(DynamoDbEnhancedClient enhancedClient,
                                       DynamoDbTable<Customer> customerTable,
                                       DynamoDbTable<MovieActor> movieActorTable) {
```

```
Customer customer2 = new Customer();
customer2.setId("2");
customer2.setEmail("cust2@example.org");

// 1. Build a batch to read from the Customer table.
ReadBatch customerBatch = ReadBatch.builder(Customer.class)
    .mappedTableResource(customerTable)
    // 1a. Specify the primary key values for the item.
    .addGetItem(b -> b.key(k ->
k.partitionValue("1").sortValue("cust1@orgname.org")))
    // 1b. Alternatively, supply a data class instances to provide the
primary key values.
    .addGetItem(customer2)
    .build();

// 2. Build a batch to read from the MovieActor table.
ReadBatch moveActorBatch = ReadBatch.builder(MovieActor.class)
    .mappedTableResource(movieActorTable)
    // 2a. Call consistentRead(Boolean.TRUE) for each item for the same
table.
    .addGetItem(b -> b.key(k ->
k.partitionValue("movie01").sortValue("actor1")))
    .consistentRead(Boolean.TRUE))
    .addGetItem(b -> b.key(k ->
k.partitionValue("movie01").sortValue("actor4")))
    .consistentRead(Boolean.TRUE))
    .build();

// 3. Add ReadBatch objects to the request.
BatchGetResultPageIterable resultPages = enhancedClient.batchGetItem(b ->
b.readBatches(customerBatch, moveActorBatch));

// 4. Retrieve the successfully requested items from each table.
resultPages.resultsForTable(customerTable).forEach(item ->
logger.info(item.toString()));
resultPages.resultsForTable(movieActorTable).forEach(item ->
logger.info(item.toString()));

// 5. Retrieve the keys of the items requested but not processed by the
service.
resultPages.forEach((BatchGetResultPage pageResult) -> {
    pageResult.unprocessedKeysForTable(customerTable).forEach(key ->
logger.info("Unprocessed item key: " + key.toString()));
    pageResult.unprocessedKeysForTable(movieActorTable).forEach(key ->
logger.info("Unprocessed item key: " + key.toString()));
});
```

```
}
```

Si supponga che i seguenti elementi siano presenti nelle due tabelle prima di eseguire il codice di esempio.

### Elementi nelle tabelle

```
Customer [id=1, name=CustName1, email=cust1@example.org,
  regDate=2023-03-31T15:46:27.688Z]
Customer [id=2, name=CustName2, email=cust2@example.org,
  regDate=2023-03-31T15:46:28.688Z]
Customer [id=3, name=CustName3, email=cust3@example.org,
  regDate=2023-03-31T15:46:29.688Z]
Customer [id=4, name=CustName4, email=cust4@example.org,
  regDate=2023-03-31T15:46:30.688Z]
Customer [id=5, name=CustName5, email=cust5@example.org,
  regDate=2023-03-31T15:46:31.689Z]
MovieActor{movieName='movie01', actorName='actor0', actingAward='actingaward0',
  actingYear=2001, actingSchoolName='null'}
MovieActor{movieName='movie01', actorName='actor1', actingAward='actingaward1',
  actingYear=2001, actingSchoolName='actingschool1'}
MovieActor{movieName='movie01', actorName='actor2', actingAward='actingaward2',
  actingYear=2001, actingSchoolName='actingschool2'}
MovieActor{movieName='movie01', actorName='actor3', actingAward='actingaward3',
  actingYear=2001, actingSchoolName='null'}
MovieActor{movieName='movie01', actorName='actor4', actingAward='actingaward4',
  actingYear=2001, actingSchoolName='actingschool4'}
```

L'output seguente mostra gli elementi restituiti e registrati dopo la riga di commento 4.

```
Customer [id=1, name=CustName1, email=cust1@example.org,
  regDate=2023-03-31T15:46:27.688Z]
Customer [id=2, name=CustName2, email=cust2@example.org,
  regDate=2023-03-31T15:46:28.688Z]
MovieActor{movieName='movie01', actorName='actor4', actingAward='actingaward4',
  actingYear=2001, actingSchoolName='actingschool4'}
MovieActor{movieName='movie01', actorName='actor1', actingAward='actingaward1',
  actingYear=2001, actingSchoolName='actingschool1'}
```

## batchWriteItem() Esempio

Il `batchWriteItem()` metodo inserisce o elimina più elementi in una o più tabelle. È possibile specificare fino a 25 singole operazioni di inserimento o eliminazione nella richiesta. L'esempio seguente utilizza le classi [ProductCataloge](#) [MovieActor](#)model mostrate in precedenza.

`WriteBatch`gli oggetti vengono creati dopo le righe di commento 1 e 2. Per la `ProductCatalog` tabella, il codice inserisce un elemento ed elimina un elemento. Per la `MovieActor` tabella dopo la riga di commento 2, il codice inserisce due elementi e ne elimina uno.

Il `batchWriteItem` metodo viene chiamato dopo la riga di commento 3. Il [builder](#) parametro fornisce le richieste batch per ogni tabella.

L'[BatchWriteResult](#) oggetto restituito fornisce metodi separati per ogni operazione per visualizzare le richieste non elaborate. Il codice dopo la riga di commento 4a fornisce le chiavi per le richieste di cancellazione non elaborate e il codice dopo la riga di commento 4b fornisce gli elementi put non elaborati.

```
public static void batchWriteItemExample(DynamoDbEnhancedClient enhancedClient,
                                         DynamoDbTable<ProductCatalog>
catalogTable,
                                         DynamoDbTable<MovieActor> movieActorTable)
{
    // 1. Build a batch to write to the ProductCatalog table.
    WriteBatch products = WriteBatch.builder(ProductCatalog.class)
        .mappedTableResource(catalogTable)
        .addPutItem(b -> b.item(getProductCatItem1()))
        .addDeleteItem(b -> b.key(k -> k
            .partitionValue(getProductCatItem2().id())
            .sortValue(getProductCatItem2().title()))
        .build();

    // 2. Build a batch to write to the MovieActor table.
    WriteBatch movies = WriteBatch.builder(MovieActor.class)
        .mappedTableResource(movieActorTable)
        .addPutItem(getMovieActorYeoh())
        .addPutItem(getMovieActorBlanchettPartial())
        .addDeleteItem(b -> b.key(k -> k
            .partitionValue(getMovieActorStreep().getMovieName())
            .sortValue(getMovieActorStreep().getActorName()))
        .build();
}
```

```

    // 3. Add WriteBatch objects to the request.
    BatchWriteResult batchWriteResult = enhancedClient.batchWriteItem(b ->
b.writeBatches(products, movies));
    // 4. Retrieve keys for items the service did not process.
    // 4a. 'unprocessedDeleteItemsForTable()' returns keys for delete requests that
did not process.
    if (batchWriteResult.unprocessedDeleteItemsForTable(movieActorTable).size() >
0) {

batchWriteResult.unprocessedDeleteItemsForTable(movieActorTable).forEach(key ->
    logger.info(key.toString()));
    }
    // 4b. 'unprocessedPutItemsForTable()' returns keys for put requests that did
not process.
    if (batchWriteResult.unprocessedPutItemsForTable(catalogTable).size() > 0) {
        batchWriteResult.unprocessedPutItemsForTable(catalogTable).forEach(key ->
            logger.info(key.toString()));
    }
}

```

I seguenti metodi di supporto forniscono gli oggetti del modello per le operazioni di inserimento ed eliminazione.

### Metodi di supporto

```

public static ProductCatalog getProductCatItem1() {
    return ProductCatalog.builder()
        .id(2)
        .isbn("1-565-85698")
        .authors(new HashSet<>(Arrays.asList("a", "b")))
        .price(BigDecimal.valueOf(30.22))
        .title("Title 55")
        .build();
}

public static ProductCatalog getProductCatItem2() {
    return ProductCatalog.builder()
        .id(4)
        .price(BigDecimal.valueOf(40.00))
        .title("Title 1")
        .build();
}

```

```
public static MovieActor getMovieActorBlanchettPartial() {
    MovieActor movieActor = new MovieActor();
    movieActor.setActorName("Cate Blanchett");
    movieActor.setMovieName("Blue Jasmine");
    movieActor.setActingYear(2023);
    movieActor.setActingAward("Best Actress");
    return movieActor;
}

public static MovieActor getMovieActorStreep() {
    MovieActor movieActor = new MovieActor();
    movieActor.setActorName("Meryl Streep");
    movieActor.setMovieName("Sophie's Choice");
    movieActor.setActingYear(1982);
    movieActor.setActingAward("Best Actress");
    movieActor.setActingSchoolName("Yale School of Drama");
    return movieActor;
}

public static MovieActor getMovieActorYeoh(){
    MovieActor movieActor = new MovieActor();
    movieActor.setActorName("Michelle Yeoh");
    movieActor.setMovieName("Everything Everywhere All at Once");
    movieActor.setActingYear(2023);
    movieActor.setActingAward("Best Actress");
    movieActor.setActingSchoolName("Royal Academy of Dance");
    return movieActor;
}
```

Si supponga che le tabelle contengano i seguenti elementi prima di eseguire il codice di esempio.

```
MovieActor{movieName='Blue Jasmine', actorName='Cate Blanchett', actingAward='Best Actress', actingYear=2013, actingSchoolName='National Institute of Dramatic Art'}
MovieActor{movieName='Sophie's Choice', actorName='Meryl Streep', actingAward='Best Actress', actingYear=1982, actingSchoolName='Yale School of Drama'}
ProductCatalog{id=4, title='Title 1', isbn='orig_isbn', authors=[b, g], price=10}
```

Al termine del codice di esempio, le tabelle contengono gli elementi seguenti.

```
MovieActor{movieName='Blue Jasmine', actorName='Cate Blanchett', actingAward='Best Actress', actingYear=2013, actingSchoolName='null'}
```



```
MovieActor{movieName='Everything Everywhere All at Once', actorName='Michelle Yeoh',
  actingAward='Best Actress', actingYear=2023, actingSchoolName='Royal Academy of
  Dance'}
ProductCatalog{id=2, title='Title 55', isbn='1-565-85698', authors=[a, b], price=30.22}
```

Si noti nella `MovieActor` tabella che l'elemento del `Blue Jasmine` filmato è stato sostituito con l'elemento utilizzato nella richiesta `put` acquisita tramite il metodo `getMovieActorBlanchettPartial()` helper. Se non è stato fornito un valore dell'attributo `data bean`, il valore nel database viene rimosso. Questo è il motivo per cui il risultato `actingSchoolName` è nullo per l'elemento del `Blue Jasmine` filmato.

### Note

Sebbene la documentazione dell'API suggerisca che è possibile utilizzare espressioni condizionali e che la capacità consumata e le metriche di raccolta possano essere restituite con richieste di immissione ed eliminazione individuali, questo non è il caso in uno scenario di scrittura in batch. Per migliorare le prestazioni delle operazioni in batch, queste singole opzioni vengono ignorate.

## Esegui operazioni di transazione

L'API `DynamoDB Enhanced Client` fornisce `transactGetItems()` i metodi e `transactWriteItems()` I metodi di transazione dell'SDK for Java forniscono atomicità, coerenza, isolamento e durabilità (ACID) nelle tabelle `DynamoDB`, aiutandoti a mantenere la correttezza dei dati nelle tue applicazioni.

### `transactGetItems()` Esempio

Il `transactGetItems()` metodo accetta fino a 100 richieste individuali di articoli. Tutti gli elementi vengono letti in un'unica transazione atomica. L'Amazon `DynamoDB Developer Guide` contiene informazioni sulle [condizioni che causano il fallimento di transactGetItems\(\) un metodo](#) e anche sul livello di isolamento utilizzato durante le chiamate. `transactGetItem()`

Dopo la riga di commento 1 nell'esempio seguente, il codice chiama il `transactGetItems()` metodo con un `builder` parametro. Il builder `addGetItem()` viene richiamato tre volte con un oggetto dati che contiene i valori chiave che l'SDK utilizzerà per generare la richiesta finale.

La richiesta restituisce un elenco di `Document` oggetti dopo la riga di commento 2. L'elenco di documenti restituito contiene istanze di `Document` non nulle dei dati degli articoli nello stesso ordine

richiesto. Il [Document.getItem\(MappedTableResource<T> mappedTableResource\)](#) metodo converte un oggetto non tipizzato in un Document oggetto Java digitato se sono stati restituiti i dati dell'elemento, altrimenti restituisce null.

```

public static void transactGetItemsExample(DynamoDbEnhancedClient enhancedClient,
                                          DynamoDbTable<ProductCatalog>
catalogTable,
                                          DynamoDbTable<MovieActor>
movieActorTable) {

    // 1. Request three items from two tables using a builder.
    final List<Document> documents = enhancedClient.transactGetItems(b -> b
        .addGetItem(catalogTable,
Key.builder().partitionValue(2).sortValue("Title 55").build())
        .addGetItem(movieActorTable, Key.builder().partitionValue("Sophie's
Choice").sortValue("Meryl Streep").build())
        .addGetItem(movieActorTable, Key.builder().partitionValue("Blue
Jasmine").sortValue("Cate Blanchett").build())
        .build());

    // 2. A list of Document objects is returned in the same order as requested.
    ProductCatalog title55 = documents.get(0).getItem(catalogTable);
    if (title55 != null) {
        logger.info(title55.toString());
    }

    MovieActor sophiesChoice = documents.get(1).getItem(movieActorTable);
    if (sophiesChoice != null) {
        logger.info(sophiesChoice.toString());
    }

    // 3. The getItem() method returns null if the Document object contains no item
    from DynamoDB.
    MovieActor blueJasmine = documents.get(2).getItem(movieActorTable);
    if (blueJasmine != null) {
        logger.info(blueJasmine.toString());
    }
}

```

Le tabelle DynamoDB contengono i seguenti elementi prima dell'esecuzione dell'esempio di codice.

```
ProductCatalog{id=2, title='Title 55', isbn='orig_isbn', authors=[b, g], price=10}
```

```
MovieActor{movieName='Sophie's Choice', actorName='Meryl Streep', actingAward='Best Actress', actingYear=1982, actingSchoolName='Yale School of Drama'}
```

Viene registrato il seguente output. Se un elemento viene richiesto ma non viene trovato, non viene restituito, come nel caso della richiesta per il film denominato `Blue Jasmine`.

```
ProductCatalog{id=2, title='Title 55', isbn='orig_isbn', authors=[b, g], price=10}
MovieActor{movieName='Sophie's Choice', actorName='Meryl Streep', actingAward='Best Actress', actingYear=1982, actingSchoolName='Yale School of Drama'}
```

## Esempi di `transactWriteItems()`

[transactWriteItems\(\)](#) Accetta fino a 100 azioni di inserimento, aggiornamento o eliminazione in una singola transazione atomica su più tabelle. La Amazon DynamoDB Developer Guide contiene dettagli sulle restrizioni e le condizioni di errore del funzionamento del servizio [DynamoDB sottostante](#).

### Esempio di base

Nell'esempio seguente, sono richieste quattro operazioni per due tabelle. Le classi del modello corrispondenti [ProductCataloge](#) [MovieActor](#) sono state mostrate in precedenza.

Ciascuna delle tre operazioni possibili (put, update ed delete) utilizza un parametro di richiesta dedicato per specificare i dettagli.

Il codice dopo la riga di commento 1 mostra la semplice variazione del metodo. `addPutItem()` Il metodo accetta un [MappedTableResource](#) oggetto e l'istanza dell'oggetto dati da inserire. L'istruzione dopo la riga di commento 2 mostra la variante che accetta un [TransactPutItemEnhancedRequest](#) istanza. Questa variante consente di aggiungere altre opzioni nella richiesta, ad esempio un'espressione di condizione. Un [esempio](#) successivo mostra un'espressione di condizione per una singola operazione.

Viene richiesta un'operazione di aggiornamento dopo la riga di commento 3.

[TransactUpdateItemEnhancedRequest](#) ha un `ignoreNulls()` metodo che consente di configurare ciò che l'SDK fa con `null` i valori sull'oggetto del modello. Se il `ignoreNulls()` metodo restituisce `true`, l'SDK non rimuove i valori degli attributi della tabella per gli attributi degli oggetti dati che sono `null`. Se il `ignoreNulls()` metodo restituisce `false`, l'SDK richiede al servizio DynamoDB di rimuovere gli attributi dall'elemento nella tabella. Il valore predefinito per `ignoreNulls` è `false`.

L'istruzione dopo la riga di commento 4 mostra la variazione di una richiesta di eliminazione che richiede un oggetto dati. Il client avanzato estrae i valori chiave prima di inviare la richiesta finale.

```

public static void transactWriteItems(DynamoDbEnhancedClient enhancedClient,
                                     DynamoDbTable<ProductCatalog> catalogTable,
                                     DynamoDbTable<MovieActor> movieActorTable) {

    enhancedClient.transactWriteItems(b -> b
        // 1. Simplest variation of put item request.
        .addPutItem(catalogTable, getProductCatId2())
        // 2. Put item request variation that accommodates condition
expressions.
        .addPutItem(movieActorTable,
TransactPutItemEnhancedRequest.builder(MovieActor.class)
            .item(getMovieActorStreep())

        .conditionExpression(Expression.builder().expression("attribute_not_exists
(movie)").build())
            .build())
        // 3. Update request that does not remove attribute values on the table
if the data object's value is null.
        .addUpdateItem(catalogTable,
TransactUpdateItemEnhancedRequest.builder(ProductCatalog.class)
            .item(getProductCatId4ForUpdate())
            .ignoreNulls(Boolean.TRUE)
            .build())
        // 4. Variation of delete request that accepts a data object. The key
values are extracted for the request.
        .addDeleteItem(movieActorTable, getMovieActorBlanchett())
    );
}

```

I seguenti metodi di supporto forniscono gli oggetti dati per i parametri. add\*Item

### Metodi di supporto

```

public static ProductCatalog getProductCatId2() {
    return ProductCatalog.builder()
        .id(2)
        .isbn("1-565-85698")
        .authors(new HashSet<>(Arrays.asList("a", "b")))
        .price(BigDecimal.valueOf(30.22))
        .title("Title 55")
        .build();
}

```

```
public static ProductCatalog getProductCatId4ForUpdate() {
    return ProductCatalog.builder()
        .id(4)
        .price(BigDecimal.valueOf(40.00))
        .title("Title 1")
        .build();
}

public static MovieActor getMovieActorBlanchett() {
    MovieActor movieActor = new MovieActor();
    movieActor.setActorName("Cate Blanchett");
    movieActor.setMovieName("Tar");
    movieActor.setActingYear(2022);
    movieActor.setActingAward("Best Actress");
    movieActor.setActingSchoolName("National Institute of Dramatic Art");
    return movieActor;
}

public static MovieActor getMovieActorStreep() {
    MovieActor movieActor = new MovieActor();
    movieActor.setActorName("Meryl Streep");
    movieActor.setMovieName("Sophie's Choice");
    movieActor.setActingYear(1982);
    movieActor.setActingAward("Best Actress");
    movieActor.setActingSchoolName("Yale School of Drama");
    return movieActor;
}
```

Le tabelle DynamoDB contengono i seguenti elementi prima dell'esecuzione dell'esempio di codice.

```
1 | ProductCatalog{id=4, title='Title 1', isbn='orig_isbn', authors=[b, g], price=10}
2 | MovieActor{movieName='Tar', actorName='Cate Blanchett', actingAward='Best Actress',
  actingYear=2022, actingSchoolName='National Institute of Dramatic Art'}
```

I seguenti elementi sono presenti nelle tabelle al termine dell'esecuzione del codice.

```
3 | ProductCatalog{id=2, title='Title 55', isbn='1-565-85698', authors=[a, b],
  price=30.22}
4 | ProductCatalog{id=4, title='Title 1', isbn='orig_isbn', authors=[b, g], price=40.0}
5 | MovieActor{movieName='Sophie's Choice', actorName='Meryl Streep', actingAward='Best
  Actress', actingYear=1982, actingSchoolName='Yale School of Drama'}
```

L'elemento sulla riga 2 è stato eliminato e le righe 3 e 5 mostrano gli elementi che sono stati inseriti. La riga 4 mostra l'aggiornamento della riga 1. Il `price` valore è l'unico valore modificato sull'elemento. Se `ignoreNulls()` avesse restituito `false`, la riga 4 sarebbe simile alla riga seguente.

```
ProductCatalog{id=4, title='Title 1', isbn='null', authors=null, price=40.0}
```

### Esempio di controllo delle condizioni

L'esempio seguente mostra l'uso di un controllo delle condizioni. Un controllo delle condizioni viene utilizzato per verificare l'esistenza di un elemento o per verificare la condizione di attributi specifici di un elemento nel database. L'articolo registrato nel controllo delle condizioni non può essere utilizzato in un'altra operazione della transazione.

#### Note

Non è possibile fare riferimento allo stesso item con diverse operazioni all'interno della stessa transazione. Ad esempio, non è possibile eseguire un controllo delle condizioni e anche tentare di aggiornare lo stesso articolo nella stessa transazione.

L'esempio mostra una per ogni tipo di operazione in una richiesta transazionale di scrittura di elementi. Dopo la riga di commento 2, il `addConditionCheck()` metodo fornisce la condizione che fallisce la transazione se il `conditionExpression` parametro restituisce un risultato positivo. `false` L'espressione di condizione restituita dal metodo mostrato nel blocco `Helper methods` verifica se l'anno di premiazione del film `Sophie's Choice` è uguale a 1982. In caso affermativo, l'espressione restituisce `false` e la transazione ha esito negativo.

Questa guida illustra in modo approfondito [le espressioni](#) in un altro argomento.

```
public static void conditionCheckFailExample(DynamoDbEnhancedClient enhancedClient,
                                             DynamoDbTable<ProductCatalog>
catalogTable,
                                             DynamoDbTable<MovieActor>
movieActorTable) {
    try {
        enhancedClient.transactWriteItems(b -> b
            // 1. Perform one of each type of operation with the next three
            methods.
```

```

        .addPutItem(catalogTable,
TransactPutItemEnhancedRequest.builder(ProductCatalog.class)
            .item(getProductCatId2()).build())
        .addUpdateItem(catalogTable,
TransactUpdateItemEnhancedRequest.builder(ProductCatalog.class)
            .item(getProductCatId4ForUpdate())
            .ignoreNulls(Boolean.TRUE).build())
        .addDeleteItem(movieActorTable,
TransactDeleteItemEnhancedRequest.builder()
            .key(b1 -> b1

.partitionValue(getMovieActorBlanchett().getMovieName())

.sortValue(getMovieActorBlanchett().getActorName())).build())
    // 2. Add a condition check on a table item that is not involved in
another operation in this request.
        .addConditionCheck(movieActorTable, ConditionCheck.builder()
            .conditionExpression(buildConditionCheckExpression())
            .key(k -> k
                .partitionValue("Sophie's Choice")
                .sortValue("Meryl Streep"))
    // 3. Specify the request to return existing values from
the item if the condition evaluates to true.
        .returnValuesOnConditionCheckFailure(ReturnValuesOnConditionCheckFailure.ALL_OLD)
            .build())
            .build());
    // 4. Catch the exception if the transaction fails and log the information.
} catch (TransactionCanceledException ex) {
    ex.cancellationReasons().stream().forEach(cancellationReason -> {
        logger.info(cancellationReason.toString());
    });
}
}

```

I seguenti metodi di supporto vengono utilizzati nell'esempio di codice precedente.

### Metodi di supporto

```

private static Expression buildConditionCheckExpression() {
    Map<String, AttributeValue> expressionValue = Map.of(
        ":year", numberValue(1982));

    return Expression.builder()

```

```

        .expression("actingyear <> :year")
        .expressionValues(expressionValue)
        .build();
    }

    public static ProductCatalog getProductCatId2() {
        return ProductCatalog.builder()
            .id(2)
            .isbn("1-565-85698")
            .authors(new HashSet<>(Arrays.asList("a", "b")))
            .price(BigDecimal.valueOf(30.22))
            .title("Title 55")
            .build();
    }

    public static ProductCatalog getProductCatId4ForUpdate() {
        return ProductCatalog.builder()
            .id(4)
            .price(BigDecimal.valueOf(40.00))
            .title("Title 1")
            .build();
    }

    public static MovieActor getMovieActorBlanchett() {
        MovieActor movieActor = new MovieActor();
        movieActor.setActorName("Cate Blanchett");
        movieActor.setMovieName("Blue Jasmine");
        movieActor.setActingYear(2013);
        movieActor.setActingAward("Best Actress");
        movieActor.setActingSchoolName("National Institute of Dramatic Art");
        return movieActor;
    }
}

```

Le tabelle DynamoDB contengono i seguenti elementi prima dell'esecuzione dell'esempio di codice.

```

1 | ProductCatalog{id=4, title='Title 1', isbn='orig_isbn', authors=[b, g], price=10}
2 | MovieActor{movieName='Sophie's Choice', actorName='Meryl Streep', actingAward='Best Actress', actingYear=1982, actingSchoolName='Yale School of Drama'}
3 | MovieActor{movieName='Tar', actorName='Cate Blanchett', actingAward='Best Actress', actingYear=2022, actingSchoolName='National Institute of Dramatic Art'}

```

I seguenti elementi sono presenti nelle tabelle al termine dell'esecuzione del codice.



```
ProductCatalog{id=4, title='Title 1', isbn='orig_isbn', authors=[b, g], price=10}
MovieActor{movieName='Sophie's Choice', actorName='Meryl Streep', actingAward='Best Actress', actingYear=1982, actingSchoolName='Yale School of Drama'}
MovieActor{movieName='Tar', actorName='Cate Blanchett', actingAward='Best Actress', actingYear=2022, actingSchoolName='National Institute of Dramatic Art'}
```

Gli elementi rimangono invariati nelle tabelle perché la transazione non è riuscita. Il `actingYear` valore del filmato `Sophie's Choice` è 1982, come mostrato nella riga 2 degli elementi della tabella prima della chiamata del `transactWriteItem()` metodo.

Per acquisire le informazioni di annullamento della transazione, racchiudi la chiamata al `transactWriteItems()` metodo in un `try` blocco e `catch` il [TransactionCanceledException](#). Dopo la riga di commento 4 dell'esempio, il codice registra ogni [CancellationReason](#) oggetto. Poiché il codice che segue la riga di commento 3 dell'esempio specifica che devono essere restituiti i valori per l'elemento che ha causato il fallimento della transazione, nel registro vengono visualizzati i valori non elaborati del database per l'elemento del `Sophie's Choice` filmato.

```
CancellationReason(Code=None)
CancellationReason(Code=None)
CancellationReason(Code=None)
CancellationReason(Item={actor=AttributeValue(S=Meryl Streep),
  movie=AttributeValue(S=Sophie's Choice), actingaward=AttributeValue(S=Best Actress),
  actingyear=AttributeValue(N=1982), actingschoolname=AttributeValue(S=Yale School of Drama)}, -,
  Code=ConditionalCheckFailed, Message=The conditional request failed.)
```

## Esempio di condizione di operazione singola

L'esempio seguente mostra l'uso di una condizione su una singola operazione in una richiesta di transazione. L'operazione di eliminazione dopo la riga di commento 1 contiene una condizione che verifica il valore dell'elemento di destinazione dell'operazione rispetto al database. In questo esempio, l'espressione condizionale creata con il metodo helper dopo la riga di commento 2 specifica che l'elemento deve essere eliminato dal database se l'anno di recitazione del film non è uguale al 2013.

[Le espressioni](#) vengono discusse più avanti in questa guida.

```
public static void singleOperationConditionFailExample(DynamoDbEnhancedClient
  enhancedClient,
```

```

DynamoDbTable<ProductCatalog> catalogTable,
                                                                    DynamoDbTable<MovieActor>
movieActorTable) {
    try {
        enhancedClient.transactWriteItems(b -> b
            .addPutItem(catalogTable,
                TransactPutItemEnhancedRequest.builder(ProductCatalog.class)
                    .item(getProductCatId2())
                    .build())
            .addUpdateItem(catalogTable,
                TransactUpdateItemEnhancedRequest.builder(ProductCatalog.class)
                    .item(getProductCatId4ForUpdate())
                    .ignoreNulls(Boolean.TRUE).build())
            // 1. Delete operation that contains a condition expression
            .addDeleteItem(movieActorTable,
                TransactDeleteItemEnhancedRequest.builder()
                    .key((Key.Builder k) -> {
                        MovieActor blanchett = getMovieActorBlanchett();
                        k.partitionValue(blanchett.getMovieName())
                            .sortValue(blanchett.getActorName());
                    })
                    .conditionExpression(buildDeleteItemExpression()))
            .returnValuesOnConditionCheckFailure(ReturnValuesOnConditionCheckFailure.ALL_OLD)
            .build())
        .build());
    } catch (TransactionCanceledException ex) {
        ex.cancellationReasons().forEach(cancellationReason ->
            logger.info(cancellationReason.toString()));
    }
}

// 2. Provide condition expression to check if 'actingyear' is not equal to 2013.
private static Expression buildDeleteItemExpression() {
    Map<String, AttributeValue> expressionValue = Map.of(
        ":year", numberValue(2013));

    return Expression.builder()
        .expression("actingyear <> :year")
        .expressionValues(expressionValue)
        .build();
}

```

I seguenti metodi di supporto vengono utilizzati nell'esempio di codice precedente.

## Metodi di supporto

```
public static ProductCatalog getProductCatId2() {
    return ProductCatalog.builder()
        .id(2)
        .isbn("1-565-85698")
        .authors(new HashSet<>(Arrays.asList("a", "b")))
        .price(BigDecimal.valueOf(30.22))
        .title("Title 55")
        .build();
}

public static ProductCatalog getProductCatId4ForUpdate() {
    return ProductCatalog.builder()
        .id(4)
        .price(BigDecimal.valueOf(40.00))
        .title("Title 1")
        .build();
}

public static MovieActor getMovieActorBlanchett() {
    MovieActor movieActor = new MovieActor();
    movieActor.setActorName("Cate Blanchett");
    movieActor.setMovieName("Blue Jasmine");
    movieActor.setActingYear(2013);
    movieActor.setActingAward("Best Actress");
    movieActor.setActingSchoolName("National Institute of Dramatic Art");
    return movieActor;
}
```

Le tabelle DynamoDB contengono i seguenti elementi prima dell'esecuzione dell'esempio di codice.

```
1 | ProductCatalog{id=4, title='Title 1', isbn='orig_isbn', authors=[b, g], price=10}
2 | MovieActor{movieName='Blue Jasmine', actorName='Cate Blanchett', actingAward='Best Actress', actingYear=2013, actingSchoolName='National Institute of Dramatic Art'}
```

I seguenti elementi sono presenti nelle tabelle al termine dell'esecuzione del codice.

```
ProductCatalog{id=4, title='Title 1', isbn='orig_isbn', authors=[b, g], price=10}
```

```
2023-03-15 11:29:07 [main] INFO org.example.tests.TransactDemoTest:168 -
  MovieActor{movieName='Blue Jasmine', actorName='Cate Blanchett', actingAward='Best
  Actress', actingYear=2013, actingSchoolName='National Institute of Dramatic Art'}
```

Gli elementi rimangono invariati nelle tabelle perché la transazione non è riuscita. Il `actingYear` valore del filmato `Blue Jasmine` è quello `2013` mostrato alla riga 2 dell'elenco degli elementi prima dell'esecuzione dell'esempio di codice.

Le seguenti righe vengono registrate nella console.

```
CancellationReason(Code=None)
CancellationReason(Code=None)
CancellationReason(Item={actor=AttributeValue(S=Cate Blanchett),
  movie=AttributeValue(S=Blue Jasmine), actingaward=AttributeValue(S=Best Actress),
  actingyear=AttributeValue(N=2013), actingschoolname=AttributeValue(S=National
  Institute of Dramatic Art)},
  Code=ConditionalCheckFailed, Message=The conditional request failed)
```

## Usa indici secondari

Gli indici secondari migliorano l'accesso ai dati definendo chiavi alternative da utilizzare nelle operazioni di interrogazione e scansione. Gli indici secondari globali (GSI) hanno una chiave di partizione e una chiave di ordinamento che possono essere diverse da quelle della tabella di base. Al contrario, gli indici secondari locali (LSI) utilizzano la chiave di partizione dell'indice primario.

Annota la classe di dati con annotazioni dell'indice secondario

Gli attributi che partecipano agli indici secondari richiedono l'annotazione `@DynamoDbSecondaryPartitionKey` o `@DynamoDbSecondarySortKey`.

La classe seguente mostra le annotazioni per due indici. Il nome `GSI SubjectLastPostedDateIndex` utilizza l'`Subject` attributo per la chiave di partizione e il `LastPostedDateTime` per la chiave di ordinamento. L'LSI denominato `ForumLastPostedDateIndex` utilizza la `ForumName` come chiave di partizione e `LastPostedDateTime` come chiave di ordinamento.

Si noti che l'`Subject` attributo ha un duplice ruolo. È la chiave di ordinamento della chiave primaria e la chiave di partizione del GSI denominato `SubjectLastPostedDateIndex`.

## Classe `MessageThread`

La `MessageThread` classe è adatta all'uso come classe di dati per la [tabella Thread di esempio](#) nella Amazon DynamoDB Developer Guide.

### Importazioni

```
import software.amazon.awssdk.enhanced.dynamodb.mapper.annotations.DynamoDbBean;
import
    software.amazon.awssdk.enhanced.dynamodb.mapper.annotations.DynamoDbPartitionKey;
import
    software.amazon.awssdk.enhanced.dynamodb.mapper.annotations.DynamoDbSecondaryPartitionKey;
import
    software.amazon.awssdk.enhanced.dynamodb.mapper.annotations.DynamoDbSecondarySortKey;
import software.amazon.awssdk.enhanced.dynamodb.mapper.annotations.DynamoDbSortKey;

import java.util.List;
```

```
@DynamoDbBean
public class MessageThread {
    private String ForumName;
    private String Subject;
    private String Message;
    private String LastPostedBy;
    private String LastPostedDateTime;
    private Integer Views;
    private Integer Replies;
    private Integer Answered;
    private List<String> Tags;

    @DynamoDbPartitionKey
    public String getForumName() {
        return ForumName;
    }

    public void setForumName(String forumName) {
        ForumName = forumName;
    }

    // Sort key for primary index and partition key for GSI
    "SubjectLastPostedDateIndex".
    @DynamoDbSortKey
    @DynamoDbSecondaryPartitionKey(indexNames = "SubjectLastPostedDateIndex")
```

```
public String getSubject() {
    return Subject;
}

public void setSubject(String subject) {
    Subject = subject;
}

// Sort key for GSI "SubjectLastPostedDateIndex" and sort key for LSI
"ForumLastPostedDateIndex".
    @DynamoDbSecondarySortKey(indexNames = {"SubjectLastPostedDateIndex",
"ForumLastPostedDateIndex"})
    public String getLastPostedDateTime() {
        return LastPostedDateTime;
    }

    public void setLastPostedDateTime(String lastPostedDateTime) {
        LastPostedDateTime = lastPostedDateTime;
    }
    public String getMessage() {
        return Message;
    }

    public void setMessage(String message) {
        Message = message;
    }

    public String getLastPostedBy() {
        return LastPostedBy;
    }

    public void setLastPostedBy(String lastPostedBy) {
        LastPostedBy = lastPostedBy;
    }

    public Integer getViews() {
        return Views;
    }

    public void setViews(Integer views) {
        Views = views;
    }

    @DynamoDbSecondaryPartitionKey(indexNames = "ForumRepliesIndex")
```

```
public Integer getReplies() {
    return Replies;
}

public void setReplies(Integer replies) {
    Replies = replies;
}

public Integer getAnswered() {
    return Answered;
}

public void setAnswered(Integer answered) {
    Answered = answered;
}

public List<String> getTags() {
    return Tags;
}

public void setTags(List<String> tags) {
    Tags = tags;
}

public MessageThread() {
    this.Answered = 0;
    this.LastPostedBy = "";
    this.ForumName = "";
    this.Message = "";
    this.LastPostedDateTime = "";
    this.Replies = 0;
    this.Views = 0;
    this.Subject = "";
}

@Override
public String toString() {
    return "MessageThread{" +
        "ForumName='" + ForumName + '\'' +
        ", Subject='" + Subject + '\'' +
        ", Message='" + Message + '\'' +
        ", LastPostedBy='" + LastPostedBy + '\'' +
        ", LastPostedDateTime='" + LastPostedDateTime + '\'' +
        ", Views=" + Views +

```

```

        ", Replies=" + Replies +
        ", Answered=" + Answered +
        ", Tags=" + Tags +
        '}';
    }
}

```

## Crea l'indice

A partire dalla versione 2.20.86 dell'SDK for Java, il `createTable()` metodo genera automaticamente indici secondari dalle annotazioni delle classi di dati. Per impostazione predefinita, tutti gli attributi della tabella di base vengono copiati in un indice e i valori di throughput assegnati sono 20 unità di capacità di lettura e 20 unità di capacità di scrittura.

Tuttavia, se si utilizza una versione SDK precedente alla 2.20.86, è necessario creare l'indice insieme alla tabella, come mostrato nell'esempio seguente. Questo esempio crea i due indici per la tabella. Thread II parametro [builder](#) dispone di metodi per configurare entrambi i tipi di indici, come illustrato dopo le righe di commento 1 e 2. Utilizzate il `indexName()` metodo del generatore di indici per associare i nomi degli indici specificati nelle annotazioni delle classi di dati al tipo di indice desiderato.

Questo codice configura tutti gli attributi della tabella in modo che finiscano in entrambi gli indici dopo le righe di commento 3 e 4. Ulteriori informazioni sulle [proiezioni degli attributi](#) sono disponibili nella [Amazon DynamoDB Developer Guide](#).

```

public static void createMessageThreadTable(DynamoDbTable<MessageThread>
messageThreadDynamoDbTable, DynamoDbClient dynamoDbClient) {
    messageThreadDynamoDbTable.createTable(b -> b
        // 1. Generate the GSI.
        .globalSecondaryIndices(gsi ->
gsi.indexName("SubjectLastPostedDateIndex")
        // 3. Populate the GSI with all attributes.
        .projection(p -> p
            .projectionType(ProjectionType.ALL))
        )
        // 2. Generate the LSI.
        .localSecondaryIndices(lsi -> lsi.indexName("ForumLastPostedDateIndex")
        // 4. Populate the LSI with all attributes.
        .projection(p -> p
            .projectionType(ProjectionType.ALL))
        )
    );
}

```



## Esegui una query utilizzando un indice

L'esempio seguente esegue una query sull'indice ForumLastPostedDateIndexsecondario locale.

Dopo la riga di commento 2, create un [QueryConditional](#) oggetto necessario per la chiamata al metodo [DynamoDbIndex.query\(\)](#).

Si ottiene un riferimento all'indice che si desidera interrogare dopo la riga di commento 3 inserendo il nome dell'indice. Dopo la riga di commento 4, si chiama il `query()` metodo sull'indice che passa l'`QueryConditional` oggetto.

È inoltre possibile configurare la query per restituire tre valori di attributo, come mostrato dopo la riga di commento 5. Se non `attributesToProject()` viene chiamata, la query restituisce tutti i valori degli attributi. Notate che i nomi degli attributi specificati iniziano con lettere minuscole. Questi nomi di attributi corrispondono a quelli utilizzati nella tabella, non necessariamente ai nomi degli attributi della classe di dati.

Dopo la riga di commento 6, scorri i risultati e registra ogni elemento restituito dalla query, inoltre memorizzalo nell'elenco per restituirlo al chiamante.

```
public static List<MessageThread> queryUsingSecondaryIndices(DynamoDbEnhancedClient
    enhancedClient,
                                                                String lastPostedDate,
                                                                DynamoDbTable<MessageThread> threadTable) {
    // 1. Log the parameter value.
    logger.info("lastPostedDate value: {}", lastPostedDate);

    // 2. Create a QueryConditional whose sort key value must be greater than or
    equal to the parameter value.
    QueryConditional queryConditional =
    QueryConditional.sortGreaterThanOrEqualTo(qc ->
        qc.partitionValue("Forum02").sortValue(lastPostedDate));

    // 3. Specify the index name to query the DynamoDbIndex instance.
    final DynamoDbIndex<MessageThread> forumLastPostedDateIndex =
    threadTable.index("ForumLastPostedDateIndex");

    // 4. Perform the query by using the QueryConditional object.
    final SdkIterable<Page<MessageThread>> pagedResult =
    forumLastPostedDateIndex.query(q -> q
        .queryConditional(queryConditional))
```

```

        // 5. Request three attribute in the results.
        .attributesToProject("forumName", "subject", "lastPostedDateTime"));

    List<MessageThread> collectedItems = new ArrayList<>();
    // 6. Iterate through the pages response and sort the items.
    pagedResult.stream().forEach(page -> page.items().stream()

    .sorted(Comparator.comparing(MessageThread::getLastPostedDateTime))
        .forEach(mt -> {
            // 7. Log the returned items and add the collection to
return to the caller.
            logger.info(mt.toString());
            collectedItems.add(mt);
        }));
    return collectedItems;
}

```

I seguenti elementi sono presenti nel database prima dell'esecuzione della query.

```

MessageThread{ForumName='Forum01', Subject='Subject01', Message='Message01',
  LastPostedBy='', LastPostedDateTime='2023.03.28', Views=0, Replies=0, Answered=0,
  Tags=null}
MessageThread{ForumName='Forum02', Subject='Subject02', Message='Message02',
  LastPostedBy='', LastPostedDateTime='2023.03.29', Views=0, Replies=0, Answered=0,
  Tags=null}
MessageThread{ForumName='Forum02', Subject='Subject04', Message='Message04',
  LastPostedBy='', LastPostedDateTime='2023.03.31', Views=0, Replies=0, Answered=0,
  Tags=null}
MessageThread{ForumName='Forum02', Subject='Subject08', Message='Message08',
  LastPostedBy='', LastPostedDateTime='2023.04.04', Views=0, Replies=0, Answered=0,
  Tags=null}
MessageThread{ForumName='Forum02', Subject='Subject10', Message='Message10',
  LastPostedBy='', LastPostedDateTime='2023.04.06', Views=0, Replies=0, Answered=0,
  Tags=null}
MessageThread{ForumName='Forum03', Subject='Subject03', Message='Message03',
  LastPostedBy='', LastPostedDateTime='2023.03.30', Views=0, Replies=0, Answered=0,
  Tags=null}
MessageThread{ForumName='Forum03', Subject='Subject06', Message='Message06',
  LastPostedBy='', LastPostedDateTime='2023.04.02', Views=0, Replies=0, Answered=0,
  Tags=null}
MessageThread{ForumName='Forum03', Subject='Subject09', Message='Message09',
  LastPostedBy='', LastPostedDateTime='2023.04.05', Views=0, Replies=0, Answered=0,
  Tags=null}

```

```

MessageThread{ForumName='Forum05', Subject='Subject05', Message='Message05',
  LastPostedBy='', LastPostedDateTime='2023.04.01', Views=0, Replies=0, Answered=0,
  Tags=null}
MessageThread{ForumName='Forum07', Subject='Subject07', Message='Message07',
  LastPostedBy='', LastPostedDateTime='2023.04.03', Views=0, Replies=0, Answered=0,
  Tags=null}

```

Le istruzioni di registrazione alle righe 1 e 6 generano il seguente output della console.

```

lastPostedDate value: 2023.03.31
MessageThread{ForumName='Forum02', Subject='Subject04', Message='', LastPostedBy='',
  LastPostedDateTime='2023.03.31', Views=0, Replies=0, Answered=0, Tags=null}
MessageThread{ForumName='Forum02', Subject='Subject08', Message='', LastPostedBy='',
  LastPostedDateTime='2023.04.04', Views=0, Replies=0, Answered=0, Tags=null}
MessageThread{ForumName='Forum02', Subject='Subject10', Message='', LastPostedBy='',
  LastPostedDateTime='2023.04.06', Views=0, Replies=0, Answered=0, Tags=null}

```

La query ha restituito elementi con un *forumName* valore di Forum02 e un *lastPostedDateTime* valore maggiore o uguale a 2023.03.31. I risultati mostrano message valori con una stringa vuota sebbene message gli attributi abbiano valori nell'indice. Questo perché l'attributo message non è stato proiettato dopo la riga di commento 5.

## Usa funzionalità di mappatura avanzate

Scopri le funzionalità avanzate dello schema delle tabelle nell'API DynamoDB Enhanced Client.

Comprendi i tipi di schemi delle tabelle

[TableSchema](#) è l'interfaccia per la funzionalità di mappatura dell'API DynamoDB Enhanced Client. Può mappare un oggetto di dati da e verso una mappa di [AttributeValues](#). Un [TableSchema](#) oggetto deve conoscere la struttura della tabella che sta mappando. Queste informazioni sulla struttura sono memorizzate in un [TableMetadata](#) oggetto.

L'API client avanzata ha diverse implementazioni di [TableSchema](#), che seguono.

Schema di tabella generato da classi annotate

È un'operazione moderatamente costosa creare una [TableSchema](#) partire da classi annotate, quindi consigliamo di farlo una sola volta, all'avvio dell'applicazione.

## [BeanTableSchema](#)

Questa implementazione è costruita sulla base degli attributi e delle annotazioni di una classe bean. Un esempio di questo approccio è illustrato nella [sezione Guida introduttiva](#).

### Note

Se a non BeanTableSchema si comporta come previsto, abilita la registrazione di debug per `per. software. amazon. awssdk. enhanced. dynamodb. beans`

## [ImmutableTableSchema](#)

Questa implementazione è costruita da una classe di dati immutabile. Questo approccio è descritto nella sezione. [???](#)

Schema di tabella generato con un builder

TableSchemaI seguenti sono creati a partire dal codice utilizzando un builder. Questo approccio è meno costoso dell'approccio che utilizza classi di dati annotate. L'approccio builder evita l'uso di annotazioni e non richiede standard di denominazione. JavaBean

## [StaticTableSchema](#)

Questa implementazione è progettata per classi di dati mutabili. La sezione introduttiva di questa guida ha dimostrato come [generare un file StaticTableSchema utilizzando un builder](#).

## [StaticImmutableTableSchema](#)

Analogamente a come si crea unStaticTableSchema, si genera un'implementazione di questo tipo TableSchema utilizzando un [builder](#) da utilizzare con classi di dati immutabili.

Schema di tabella per dati senza schema fisso

## [DocumentTableSchema](#)

A differenza di altre implementazioni diTableSchema, non si definiscono gli attributi per unDocumentTableSchemaistanza. Di solito, si specificano solo chiavi primarie e fornitori di convertitori di attributi. Un'EnhancedDocumentistanza fornisce gli attributi creati da singoli elementi o da una stringa JSON.

## Includi o escludi gli attributi in modo esplicito

L'API DynamoDB Enhanced Client offre annotazioni per escludere gli attributi delle classi di dati dal diventare attributi su una tabella. Con l'API, puoi anche utilizzare un nome di attributo diverso dal nome dell'attributo della classe di dati.

### Escludi gli attributi

Per ignorare gli attributi che non devono essere mappati su una tabella DynamoDB, contrassegna l'attributo con l'annotazione. `@DynamoDbIgnore`

```
private String internalKey;

@dynamoDbIgnore
public String getInternalKey() { return this.internalKey; }
public void setInternalKey(String internalKey) { return this.internalKey =
    internalKey;}
```

### Includi gli attributi

Per modificare il nome di un attributo utilizzato nella tabella DynamoDB, contrassegna con `@DynamoDbAttribute` l'annotazione e fornisci un nome diverso.

```
private String internalKey;

@dynamoDbAttribute("renamedInternalKey")
public String getInternalKey() { return this.internalKey; }
public void setInternalKey(String internalKey) { return this.internalKey =
    internalKey;}
```

## Controlla la conversione degli attributi

Per impostazione predefinita, uno schema tabellare fornisce convertitori per molti tipi Java comuni tramite un'implementazione predefinita dell'[AttributeConverterProvider](#) interfaccia. È possibile modificare il comportamento predefinito generale con un'[AttributeConverterProvider](#) implementazione personalizzata. È inoltre possibile modificare il convertitore per un singolo attributo.

Per un elenco dei convertitori disponibili, consulta l'[AttributeConverter](#) interfaccia Java doc.

## Fornisci fornitori di convertitori di attributi personalizzati

È possibile fornire una singola `AttributeConverterProvider` o una catena di messaggi ordinati `AttributeConverterProvider` tramite l'`@DynamoDbBean(converterProviders = {...})` annotazione. Qualsiasi personalizzazione `AttributeConverterProvider` deve estendere l'`AttributeConverterProvider` interfaccia.

Tieni presente che se fornisci la tua catena di fornitori di convertitori di attributi, sostituirai il provider di convertitori predefinito, `DefaultAttributeConverterProvider`. Se si desidera utilizzare la funzionalità di `DefaultAttributeConverterProvider`, è necessario includerla nella catena.

È anche possibile annotare il bean con un array `{}` vuoto. Ciò disabilita l'uso di qualsiasi fornitore di convertitori di attributi, incluso quello predefinito. In questo caso tutti gli attributi che devono essere mappati devono avere un proprio convertitore di attributi.

Il frammento seguente mostra un singolo fornitore di convertitori.

```
@DynamoDbBean(converterProviders = ConverterProvider1.class)
public class Customer {

}
```

Il seguente frammento mostra l'uso di una catena di fornitori di convertitori. Poiché l'SDK predefinito viene fornito per ultimo, ha la priorità più bassa.

```
@DynamoDbBean(converterProviders = {
    ConverterProvider1.class,
    ConverterProvider2.class,
    DefaultAttributeConverterProvider.class})
public class Customer {

}
```

I generatori di schemi di tabelle statiche hanno un `attributeConverterProviders()` metodo che funziona allo stesso modo. Questo è mostrato nel frammento seguente.

```
private static final StaticTableSchema<Customer> CUSTOMER_TABLE_SCHEMA =
    StaticTableSchema.builder(Customer.class)
        .newItemSupplier(Customer::new)
        .addAttribute(String.class, a -> a.name("name")
            a.getter(Customer::getName)
```

```

        a.setter(Customer::setName))
    .attributeConverterProviders(converterProvider1, converterProvider2)
    .build();

```

## Sovrascrivi la mappatura di un singolo attributo

Per sovrascrivere il modo in cui viene mappato un singolo attributo, fornisci un `AttributeConverter` per l'attributo. Questa aggiunta sostituisce tutti i convertitori forniti `AttributeConverterProviders` nello schema della tabella. Questo aggiunge un convertitore personalizzato solo per quell'attributo. Altri attributi, anche quelli dello stesso tipo, non utilizzeranno quel convertitore a meno che non sia specificato esplicitamente per quegli altri attributi.

L'`@DynamoDbConvertedBy` annotazione viene utilizzata per specificare la `AttributeConverter` classe personalizzata, come mostrato nel frammento seguente.

```

@dynamoDbBean
public class Customer {
    private String name;

    @DynamoDbConvertedBy(CustomAttributeConverter.class)
    public String getName() { return this.name; }
    public void setName(String name) { this.name = name;}
}

```

I costruttori di schemi statici hanno un metodo di creazione di attributi equivalente.

`attributeConverter()` Questo metodo accetta un'istanza di un `AttributeConverter` come illustrato di seguito.

```

private static final StaticTableSchema<Customer> CUSTOMER_TABLE_SCHEMA =
    StaticTableSchema.builder(Customer.class)
        .newItemSupplier(Customer::new)
        .addAttribute(String.class, a -> a.name("name")
            a.getter(Customer::getName)
            a.setter(Customer::setName)
            a.attributeConverter(customAttributeConverter))
        .build();

```

## Esempio

Questo esempio mostra un'`AttributeConverterProvider` implementazione che fornisce un convertitore di attributi per [java.net.HttpCookie](http://java.net/HttpCookie) oggetti.

La `SimpleUser` classe seguente contiene un attributo denominato `lastUsedCookie` che è un'istanza di `HttpCookie`.

Il parametro delle `@DynamoDbBean` annotazioni elenca le due `AttributeConverterProvider` classi che forniscono convertitori.

### Class with annotations

```
@DynamoDbBean(converterProviders = {CookieConverterProvider.class,
DefaultAttributeConverterProvider.class})
public static final class SimpleUser {
    private String name;
    private HttpCookie lastUsedCookie;

    @DynamoDbPartitionKey
    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public HttpCookie getLastUsedCookie() {
        return lastUsedCookie;
    }

    public void setLastUsedCookie(HttpCookie lastUsedCookie) {
        this.lastUsedCookie = lastUsedCookie;
    }
}
```

### Static table schema

```
private static final TableSchema<SimpleUser> SIMPLE_USER_TABLE_SCHEMA =
    TableSchema.builder(SimpleUser.class)
        .newItemSupplier(SimpleUser::new)
        .attributeConverterProviders(CookieConverterProvider.create(),
AttributeConverterProvider.defaultProvider())
        .addAttribute(String.class, a -> a.name("name")
            .setter(SimpleUser::setName)
            .getter(SimpleUser::getName)
            .tags(StaticAttributeTags.primaryPartitionKey()))
        .addAttribute(HttpCookie.class, a -> a.name("lastUsedCookie"))
```



```

        .setter(SimpleUser::setLastUsedCookie)
        .getter(SimpleUser::getLastUsedCookie))
    .build();

```

`CookieConverterProvider` Nell'esempio seguente viene fornita un'istanza di un `HttpCookieConverter`

```

public static final class CookieConverterProvider implements
AttributeConverterProvider {
    private final Map<EnhancedType<?>, AttributeConverter<?>> converterCache =
ImmutableMap.of(
    // 1. Add HttpCookieConverter to the internal cache.
    EnhancedType.of(HttpCookie.class), new HttpCookieConverter());

    public static CookieConverterProvider create() {
        return new CookieConverterProvider();
    }

    // The SDK calls this method to find out if the provider contains a
AttributeConverter instance
    // for the EnhancedType<T> argument.
    @SuppressWarnings("unchecked")
    @Override
    public <T> AttributeConverter<T> converterFor(EnhancedType<T> enhancedType) {
        return (AttributeConverter<T>) converterCache.get(enhancedType);
    }
}

```

Codice di conversione

Nel `transformFrom()` metodo della `HttpCookieConverter` classe seguente, il codice riceve un'istanza `HttpCookie` e la trasforma in una mappa DynamoDB memorizzata come attributo.

Il `transformTo()` metodo riceve un parametro di mappa DynamoDB, quindi richiama `HttpCookie` il costruttore che richiede un nome e un valore.

```

public static final class HttpCookieConverter implements
AttributeConverter<HttpCookie> {

    @Override
    public AttributeValue transformFrom(HttpCookie httpCookie) {

```

```

        return AttributeValue.fromM(
            Map.of ("cookieName", AttributeValue.fromS(httpCookie.getName()),
                "cookieValue", AttributeValue.fromS(httpCookie.getValue()))
        );
    }

    @Override
    public HttpCookie transformTo(AttributeValue attributeValue) {
        Map<String, AttributeValue> map = attributeValue.m();
        return new HttpCookie(
            map.get("cookieName").s(),
            map.get("cookieValue").s());
    }

    @Override
    public EnhancedType<HttpCookie> type() {
        return EnhancedType.of(HttpCookie.class);
    }

    @Override
    public AttributeValueType attributeValueType() {
        return AttributeValueType.M;
    }
}

```

## Modifica il comportamento di aggiornamento degli attributi

È possibile personalizzare il comportamento di aggiornamento dei singoli attributi quando si esegue un'operazione di aggiornamento. [Alcuni esempi di operazioni di aggiornamento nell'API DynamoDB Enhanced Client sono `updateItem\(\)` e `transactWriteItems\(\)`](#)

Ad esempio, immaginate di voler memorizzare un timestamp creato su un vostro record. Tuttavia, desiderate che il suo valore venga scritto solo se non esiste alcun valore esistente per l'attributo già nel database. In questo caso, si utilizza il comportamento di [WRITE\\_IF\\_NOT\\_EXISTS](#) aggiornamento.

L'esempio seguente mostra l'annotazione che aggiunge il comportamento all'`createdOn` attributo.

```

@DynamoDbBean
public class Customer extends GenericRecord {
    private String id;
    private Instant createdOn;
}

```

```

@DynamoDbPartitionKey
public String getId() { return this.id; }
public void setId(String id) { this.name = id; }

@DynamoDbUpdateBehavior(UpdateBehavior.WRITE_IF_NOT_EXISTS)
public Instant getCreatedOn() { return this.createdOn; }
public void setCreatedOn(Instant createdOn) { this.createdOn = createdOn; }
}

```

È possibile dichiarare lo stesso comportamento di aggiornamento quando si crea uno schema di tabella statico, come mostrato nell'esempio seguente dopo la riga di commento 1.

```

static final TableSchema<Customer> CUSTOMER_TABLE_SCHEMA =
    TableSchema.builder(Customer.class)
        .newItemSupplier(Customer::new)
        .addAttribute(String.class, a -> a.name("id")
            .getter(Customer::getId)
            .setter(Customer::setId)

        .tags(StaticAttributeTags.primaryPartitionKey()))
        .addAttribute(Instant.class, a -> a.name("createdOn")
            .getter(Customer::getCreatedOn)
            .setter(Customer::setCreatedOn)
            // 1. Add an UpdateBehavior.

        .tags(StaticAttributeTags.updateBehavior(UpdateBehavior.WRITE_IF_NOT_EXISTS)))
        .build();

```

## Appiattisci gli attributi di altre classi

Se gli attributi della tabella sono distribuiti tra diverse classi Java, tramite ereditarietà o composizione, l'API Enhanced Client di DynamoDB fornisce supporto per riunire gli attributi in un'unica classe.

### Usa l'ereditarietà

Se le tue classi utilizzano l'ereditarietà, usa i seguenti approcci per appiattare la gerarchia.

### Usa fagioli annotati

Per l'approccio all'annotazione, entrambe le classi devono contenere l'@DynamoDbBeanannotazione e una classe deve contenere una o più annotazioni chiave primarie.

Di seguito vengono illustrati esempi di classi di dati che hanno una relazione di ereditarietà.

## Standard data class

```

@DynamoDbBean
public class Customer extends GenericRecord {
    private String name;

    public String getName() { return name; }
    public void setName(String name) { this.name = name; }
}

@DynamoDbBean
public abstract class GenericRecord {
    private String id;
    private String createdAt;

    @DynamoDbPartitionKey
    public String getId() { return id; }
    public void setId(String id) { this.id = id; }

    public String getCreatedAt() { return createdAt; }
    public void setCreatedAt(String createdAt) { this.createdAt =
createdAt; }
}

```

## Lombok

[onMethodL'opzione](#) di Lombok copia le annotazioni DynamoDB basate sugli attributi, ad esempio, sul codice generato. `@DynamoDbPartitionKey`

```

@DynamoDbBean
@Data
@ToString(callSuper = true)
public class Customer extends GenericRecord {
    private String name;
}

@Data
@DynamoDbBean
public abstract class GenericRecord {
    @Getter(onMethod_=@DynamoDbPartitionKey)
    private String id;
    private String createdAt;
}

```

## Usa schemi statici

Per l'approccio allo schema statico, utilizzate il `extend()` metodo del generatore per comprimere gli attributi della classe principale nella classe figlia. Questo è mostrato dopo la riga di commento 1 nell'esempio seguente.

```

        StaticTableSchema<org.example.tests.model.inheritance.stat.GenericRecord>
GENERIC_RECORD_SCHEMA =

StaticTableSchema.builder(org.example.tests.model.inheritance.stat.GenericRecord.class)
    // The partition key will be inherited by the top level mapper.
    .addAttribute(String.class, a -> a.name("id"))

    .getter(org.example.tests.model.inheritance.stat.GenericRecord::getId)

    .setter(org.example.tests.model.inheritance.stat.GenericRecord::setId)
        .tags(primaryPartitionKey()))
    .addAttribute(String.class, a -> a.name("created_date"))

    .getter(org.example.tests.model.inheritance.stat.GenericRecord::getCreatedDate)

    .setter(org.example.tests.model.inheritance.stat.GenericRecord::setCreatedDate))
    .build();

        StaticTableSchema<org.example.tests.model.inheritance.stat.Customer>
CUSTOMER_SCHEMA =

StaticTableSchema.builder(org.example.tests.model.inheritance.stat.Customer.class)

    .newItemSupplier(org.example.tests.model.inheritance.stat.Customer::new)
        .addAttribute(String.class, a -> a.name("name"))

    .getter(org.example.tests.model.inheritance.stat.Customer::getName)

    .setter(org.example.tests.model.inheritance.stat.Customer::setName))
    // 1. Use the extend() method to collapse the parent attributes
onto the child class.
        .extend(GENERIC_RECORD_SCHEMA) // All the attributes of the
GenericRecord schema are added to Customer.
    .build();

```

Il precedente esempio di schema statico utilizza le seguenti classi di dati. Poiché la mappatura viene definita quando si crea lo schema statico della tabella, le classi di dati non richiedono annotazioni.

## Classi di dati

### Standard data class

```
public class Customer extends GenericRecord {
    private String name;

    public String getName() { return name; }
    public void setName(String name) { this.name = name; }
}

public abstract class GenericRecord {
    private String id;
    private String createdAt;

    public String getId() { return id; }
    public void setId(String id) { this.id = id; }

    public String getCreatedAt() { return createdAt; }
    public void setCreatedAt(String createdAt) { this.createdAt =
    createdAt; }
```

### Lombok

```
@Data
@ToString(callSuper = true)
public class Customer extends GenericRecord{
    private String name;
}

@Data
public abstract class GenericRecord {
    private String id;
    private String createdAt;
}
```

### Usa la composizione

Se le tue classi usano la composizione, usa i seguenti approcci per appiattare la gerarchia.

## Usa fagioli annotati

L'`@DynamoDbFlatten` annotazione appiattisce la classe contenuta.

I seguenti esempi di classi di dati utilizzano l'`@DynamoDbFlatten` annotazione per aggiungere efficacemente tutti gli attributi della `GenericRecord` classe contenuta alla classe. `Customer`

### Standard data class

```
@DynamoDbBean
public class Customer {
    private String name;
    private GenericRecord record;

    public String getName() { return this.name; }
    public void setName(String name) { this.name = name; }

    @DynamoDbFlatten
    public GenericRecord getRecord() { return this.record; }
    public void setRecord(GenericRecord record) { this.record = record; }

    @DynamoDbBean
    public class GenericRecord {
        private String id;
        private String createdAt;

        @DynamoDbPartitionKey
        public String getId() { return this.id; }
        public void setId(String id) { this.id = id; }

        public String getCreatedAt() { return this.createdAt; }
        public void setCreatedAt(String createdAt) { this.createdAt =
        createdAt; }
    }
}
```

### Lombok

```
@Data
@dynamoDbBean
public class Customer {
    private String name;
    @Getter(onMethod_=@DynamoDbFlatten)
    private GenericRecord record;
}
```

```

}

@Data
@DynamoDbBean
public class GenericRecord {
    @Getter(onMethod_=@DynamoDbPartitionKey)
    private String id;
    private String createdAt;
}

```

Puoi usare l'annotazione `flatten` per appiattire tutte le diverse classi idonee di cui hai bisogno. Vengono applicati i vincoli seguenti:

- Tutti i nomi degli attributi devono essere unici dopo essere stati appiattiti.
- Non deve mai esserci più di una chiave di partizione, chiave di ordinamento o nome di tabella.

### Usa schemi statici

Quando crei uno schema di tabella statico, usa il `flatten()` metodo del generatore. Fornite anche i metodi `getter` e `setter` che identificano la classe contenuta.

```

StaticTableSchema<GenericRecord> GENERIC_RECORD_SCHEMA =
    StaticTableSchema.builder(GenericRecord.class)
        .newItemSupplier(GenericRecord::new)
        .addAttribute(String.class, a -> a.name("id")
            .getter(GenericRecord::getId)
            .setter(GenericRecord::setId)
            .tags(primaryPartitionKey()))
        .addAttribute(String.class, a -> a.name("created_date")
            .getter(GenericRecord::getCreatedAt)
            .setter(GenericRecord::setCreatedAt))
        .build();

StaticTableSchema<Customer> CUSTOMER_SCHEMA =
    StaticTableSchema.builder(Customer.class)
        .newItemSupplier(Customer::new)
        .addAttribute(String.class, a -> a.name("name")
            .getter(Customer::getName)
            .setter(Customer::setName))
        // Because we are flattening a component object, we supply a
getter and setter so the

```



```
        // mapper knows how to access it.
        .flatten(GENERIC_RECORD_SCHEMA, Customer::getRecord,
Customer::setRecord)
        .build();
```

Il precedente esempio di schema statico utilizza le seguenti classi di dati.

## Classi di dati

### Standard data class

```
public class Customer {
    private String name;
    private GenericRecord record;

    public String getName() { return this.name; }
    public void setName(String name) { this.name = name; }

    public GenericRecord getRecord() { return this.record; }
    public void setRecord(GenericRecord record) { this.record = record; }

public class GenericRecord {
    private String id;
    private String createdAt;

    public String getId() { return this.id; }
    public void setId(String id) { this.id = id; }

    public String getCreatedAt() { return this.createdAt; }
    public void setCreatedAt(String createdAt) { this.createdAt =
createdAt; }
}
```

## Lombok

```
@Data
public class Customer {
    private String name;
    private GenericRecord record;
}

@Data
public class GenericRecord {
```

```
private String id;
private String createdAt;
}
```

Puoi utilizzare il modello builder per appiattare tutte le diverse classi idonee di cui hai bisogno.

### Implicazioni per altro codice

Quando si utilizza l'`@DynamoDbFlatten` attributo (o il metodo `flatten()` builder), l'elemento in DynamoDB contiene un attributo per ogni attributo dell'oggetto composto. Include anche gli attributi dell'oggetto che lo compone.

Al contrario, se annotate una classe di dati con una classe composta e non la utilizzate `@DynamoDbFlatten`, l'elemento viene salvato con l'oggetto composto come attributo singolo.

Ad esempio, confrontate la `Customer` classe mostrata nell'esempio di [appiattimento con l'esempio di composizione con](#) e senza appiattimento dell'attributo. record È possibile visualizzare la differenza con JSON come illustrato nella tabella seguente.

Con appiattimento	Senza appiattimento
3 attributi	2 attributi
<pre>{   "id": "1",   "createdAt": "today",   "name": "my name" }</pre>	<pre>{   "id": "1",   "record": {     "createdAt": "today",     "name": "my name"   } }</pre>

La differenza diventa importante se si dispone di altro codice che accede alla tabella DynamoDB che prevede di trovare determinati attributi.

### Lavora con attributi annidati

Un attributo annidato in DynamoDB è incorporato in un altro attributo. Alcuni esempi sono gli elementi dell'elenco e le voci della mappa.

In Java, un attributo annidato di DynamoDB corrisponde a un membro di una classe che è un o. List Map Corrisponde anche a un'istanza di tipo complesso, ad esempio Address oPhoneNumber, utilizzata nella classe seguente. Person

## Classe **Person**

```
@DynamoDbBean
public class Person {
    Integer id;
    String firstName;
    String lastName;
    Integer age;
    Map<String, Address> addresses;
    List<PhoneNumber> phoneNumbers;

    List<String> hobbies;

    @DynamoDbPartitionKey
    public Integer getId() {
        return id;
    }

    public void setId(Integer id) {
        this.id = id;
    }

    public String getFirstName() {
        return firstName;
    }

    public void setFirstName(String firstName) {
        this.firstName = firstName;
    }

    public String getLastName() {
        return lastName;
    }

    public void setLastName(String lastName) {
        this.lastName = lastName;
    }

    public Integer getAge() {
```

```
        return age;
    }

    public void setAge(Integer age) {
        this.age = age;
    }

    public Map<String, Address> getAddresses() {
        return addresses;
    }

    public void setAddresses(Map<String, Address> addresses) {
        this.addresses = addresses;
    }

    public List<PhoneNumber> getPhoneNumbers() {
        return phoneNumbers;
    }

    public void setPhoneNumbers(List<PhoneNumber> phoneNumbers) {
        this.phoneNumbers = phoneNumbers;
    }

    public List<String> getHobbies() {
        return hobbies;
    }

    public void setHobbies(List<String> hobbies) {
        this.hobbies = hobbies;
    }

    @Override
    public String toString() {
        return "Person{" +
            "id=" + id +
            ", firstName='" + firstName + '\'' +
            ", lastName='" + lastName + '\'' +
            ", age=" + age +
            ", addresses=" + addresses +
            ", phoneNumbers=" + phoneNumbers +
            ", hobbies=" + hobbies +
            '}';
    }
}
```

```
}
```

## Classe **Address**

```
@DynamoDbBean
public class Address {
    private String street;
    private String city;
    private String state;
    private String zipCode;

    public Address() {
    }

    public String getStreet() {
        return this.street;
    }

    public String getCity() {
        return this.city;
    }

    public String getState() {
        return this.state;
    }

    public String getZipCode() {
        return this.zipCode;
    }

    public void setStreet(String street) {
        this.street = street;
    }

    public void setCity(String city) {
        this.city = city;
    }

    public void setState(String state) {
        this.state = state;
    }

    public void setZipCode(String zipCode) {
```

```
        this.zipCode = zipCode;
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
        Address address = (Address) o;
        return Objects.equals(street, address.street) && Objects.equals(city,
address.city) && Objects.equals(state, address.state) && Objects.equals(zipCode,
address.zipCode);
    }

    @Override
    public int hashCode() {
        return Objects.hash(street, city, state, zipCode);
    }

    @Override
    public String toString() {
        return "Address{" +
            "street='" + street + '\'' +
            ", city='" + city + '\'' +
            ", state='" + state + '\'' +
            ", zipCode='" + zipCode + '\'' +
            '}';
    }
}
```

## Classe **PhoneNumber**

```
@DynamoDbBean
public class PhoneNumber {
    String type;
    String number;

    public String getType() {
        return type;
    }

    public void setType(String type) {
        this.type = type;
    }
}
```

```

public String getNumber() {
    return number;
}

public void setNumber(String number) {
    this.number = number;
}

@Override
public String toString() {
    return "PhoneNumber{" +
        "type='" + type + '\'' +
        ", number='" + number + '\'' +
        '}';
}
}

```

## Mappa gli attributi annidati

### Usa classi annotate

È possibile salvare gli attributi annidati per le classi personalizzate annotandole. La `Address` classe e la `PhoneNumber` classe mostrate in precedenza vengono annotate solo con l'annotazione. `@DynamoDbBean` Quando l'API `DynamoDB Enhanced Client` crea lo schema della tabella per la classe con `Person` il seguente frammento, l'API rileva l'uso delle classi `PhoneNumber` and e crea le mappature corrispondenti per `Address` funzionare con `DynamoDB`.

```
TableSchema<Person> personTableSchema = TableSchema.fromBean(Person.class);
```

### Usa schemi annidati

L'approccio alternativo consiste nell'utilizzare generatori di schemi di tabelle statiche per ciascuna delle classi, come mostrato nel codice seguente.

Gli schemi di tabella per le `PhoneNumber` classi `Address` and sono astratti, nel senso che non possono essere utilizzati con una tabella `DynamoDB`. Questo perché mancano di definizioni per la chiave primaria. Vengono tuttavia utilizzati come schemi annidati nello schema tabellare della classe. `Person`

Dopo le righe di commento 1 e 2 nella definizione di `PERSON_TABLE_SCHEMA`, viene visualizzato il codice che utilizza gli schemi di tabella astratti. L'uso di `documentOf` nel

`EnhanceType.documentOf(...)` metodo non indica che il metodo restituisca un `EnhancedDocument` tipo di `Enhanced Document API`. Il `documentOf(...)` metodo in questo contesto restituisce un oggetto che sa come mappare l'argomento della classe da e verso gli attributi della tabella `DynamoDB` utilizzando l'argomento dello schema della tabella.

Codice dello schema statico

```
// Abstract table schema that cannot be used to work with a DynamoDB table,
// but can be used as a nested schema.
public static final TableSchema<Address> TABLE_SCHEMA_ADDRESS =
TableSchema.builder(Address.class)
    .newItemSupplier(Address::new)
    .addAttribute(String.class, a -> a.name("street")
        .getter(Address::getStreet)
        .setter(Address::setStreet))
    .addAttribute(String.class, a -> a.name("city")
        .getter(Address::getCity)
        .setter(Address::setCity))
    .addAttribute(String.class, a -> a.name("zipcode")
        .getter(Address::getZipCode)
        .setter(Address::setZipCode))
    .addAttribute(String.class, a -> a.name("state")
        .getter(Address::getState)
        .setter(Address::setState))
    .build();

// Abstract table schema that cannot be used to work with a DynamoDB table,
// but can be used as a nested schema.
public static final TableSchema<PhoneNumber> TABLE_SCHEMA_PHONENUMBER =
TableSchema.builder(PhoneNumber.class)
    .newItemSupplier(PhoneNumber::new)
    .addAttribute(String.class, a -> a.name("type")
        .getter(PhoneNumber::getType)
        .setter(PhoneNumber::setType))
    .addAttribute(String.class, a -> a.name("number")
        .getter(PhoneNumber::getNumber)
        .setter(PhoneNumber::setNumber))
    .build();

// A static table schema that can be used with a DynamoDB table.
// The table schema contains two nested schemas that are used to perform mapping
to/from DynamoDB.
public static final TableSchema<Person> PERSON_TABLE_SCHEMA =
```



```

TableSchema.builder(Person.class)
    .newItemSupplier(Person::new)
    .addAttribute(Integer.class, a -> a.name("id")
        .getter(Person::getId)
        .setter(Person::setId)
        .addTag(StaticAttributeTags.primaryPartitionKey()))
    .addAttribute(String.class, a -> a.name("firstName")
        .getter(Person::getFirstName)
        .setter(Person::setFirstName))
    .addAttribute(String.class, a -> a.name("lastName")
        .getter(Person::getLastName)
        .setter(Person::setLastName))
    .addAttribute(Integer.class, a -> a.name("age")
        .getter(Person::getAge)
        .setter(Person::setAge))
    .addAttribute(EnhancedType.listOf(String.class), a ->
a.name("hobbies")
        .getter(Person::getHobbies)
        .setter(Person::setHobbies))
    .addAttribute(EnhancedType.mapOf(
        EnhancedType.of(String.class),
        // 1. Use mapping functionality of the Address table
schema.
        EnhancedType.documentOf(Address.class,
TABLE_SCHEMA_ADDRESS)), a -> a.name("addresses")
        .getter(Person::getAddresses)
        .setter(Person::setAddresses))
    .addAttribute(EnhancedType.listOf(
        // 2. Use mapping functionality of the PhoneNumber table
schema.
        EnhancedType.documentOf(PhoneNumber.class,
TABLE_SCHEMA_PHONENUMBER)), a -> a.name("phoneNumbers")
        .getter(Person::getPhoneNumbers)
        .setter(Person::setPhoneNumbers))
    .build();

```

## Attributi annidati del progetto

Per `query()` i `scan()` metodi, è possibile specificare quali attributi si desidera vengano restituiti nei risultati utilizzando chiamate di metodo come `addNestedAttributeToProject()` e `attributesToProject()`. L'API DynamoDB Enhanced Client converte i parametri di chiamata al metodo Java [in espressioni di proiezione](#) prima dell'invio della richiesta.

L'esempio seguente popola la `Person` tabella con due elementi, quindi esegue tre operazioni di scansione.

La prima scansione accede a tutti gli elementi della tabella per confrontare i risultati con le altre operazioni di scansione.

La seconda scansione utilizza il metodo `addNestedAttributeToProject()` builder per restituire solo il `street` valore dell'attributo.

La terza operazione di scansione utilizza il metodo `attributesToProject()` builder per restituire i dati per l'attributo di primo livello, `hobbies`. Il tipo di attributo di `hobbies` è un elenco. Per accedere a singoli elementi dell'elenco, eseguire un' `get()` operazione sull'elenco.

```
personDynamoDbTable = getDynamoDbEnhancedClient().table("Person",
PERSON_TABLE_SCHEMA);
PersonUtils.createPersonTable(personDynamoDbTable, getDynamoDbClient());
// Use a utility class to add items to the Person table.
List<Person> personList = PersonUtils.getItemsForCount(2);
// This utility method performs a put against DynamoDB to save the instances in
the list argument.
PersonUtils.putCollection(getDynamoDbEnhancedClient(), personList,
personDynamoDbTable);

// The first scan logs all items in the table to compare to the results of the
subsequent scans.
final PageIterable<Person> allItems = personDynamoDbTable.scan();
allItems.items().forEach(p ->
    // 1. Log what is in the table.
    logger.info(p.toString()));

// Scan for nested attributes.
PageIterable<Person> streetScanResult = personDynamoDbTable.scan(b -> b
    // Use the 'addNestedAttributeToProject()' or
'addNestedAttributesToProject()' to access data nested in maps in DynamoDB.
    .addNestedAttributeToProject(
        NestedAttributeName.create("addresses", "work", "street")
    ));

streetScanResult.items().forEach(p ->
    //2. Log the results of requesting nested attributes.
    logger.info(p.toString()));

// Scan for a top-level list attribute.
```

```

    PageIterable<Person> phoneNumbersScanResult = personDynamoDbTable.scan(b -> b
        // Use the 'attributesToProject()' method to access first-level
attributes.
        .attributesToProject("hobbies"));

    phoneNumbersScanResult.items().forEach((p) -> {
        // 3. Log the results of the request for the 'hobbies' attribute.
        logger.info(p.toString());
        // To access an item in a list, first get the parent attribute, 'hobbies',
then access items in the list.
        String hobby = p.getHobbies().get(1);
        // 4. Log an item in the list.
        logger.info(hobby);
    });

```

```

// Logged results from comment line 1.
Person{id=2, firstName='first name 2', lastName='last name 2', age=11,
  addresses={work=Address{street='street 21', city='city 21', state='state 21',
zipCode='33333'}, home=Address{street='street 2', city='city 2', state='state 2',
zipCode='22222'}}, phoneNumbers=[PhoneNumber{type='home', number='222-222-2222'},
PhoneNumber{type='work', number='333-333-3333'}], hobbies=[hobby 2, hobby 21]}
Person{id=1, firstName='first name 1', lastName='last name 1', age=11,
  addresses={work=Address{street='street 11', city='city 11', state='state 11',
zipCode='22222'}, home=Address{street='street 1', city='city 1', state='state 1',
zipCode='11111'}}, phoneNumbers=[PhoneNumber{type='home', number='111-111-1111'},
PhoneNumber{type='work', number='222-222-2222'}], hobbies=[hobby 1, hobby 11]}

// Logged results from comment line 2.
Person{id=null, firstName='null', lastName='null', age=null,
  addresses={work=Address{street='street 21', city='null', state='null',
zipCode='null'}}}, phoneNumbers=null, hobbies=null}
Person{id=null, firstName='null', lastName='null', age=null,
  addresses={work=Address{street='street 11', city='null', state='null',
zipCode='null'}}}, phoneNumbers=null, hobbies=null}

// Logged results from comment lines 3 and 4.
Person{id=null, firstName='null', lastName='null', age=null, addresses=null,
  phoneNumbers=null, hobbies=[hobby 2, hobby 21]}
hobby 21
Person{id=null, firstName='null', lastName='null', age=null, addresses=null,
  phoneNumbers=null, hobbies=[hobby 1, hobby 11]}
hobby 11

```

**Note**

Se il `attributesToProject()` metodo segue qualsiasi altro metodo di creazione che aggiunge gli attributi che desiderate proiettare, l'elenco dei nomi degli attributi fornito `attributesToProject()` sostituisce tutti gli altri nomi di attributo.

Una scansione eseguita con l'`ScanEnhancedRequest` istanza nel frammento seguente restituisce solo dati relativi agli hobby.

```
ScanEnhancedRequest lastOverwrites = ScanEnhancedRequest.builder()
    .addNestedAttributeToProject(
        NestedAttributeName.create("addresses", "work", "street"))
    .addAttributeToProject("firstName")
    // If the 'attributesToProject()' method follows other builder methods
    that add attributes for projection,
    // its list of attributes replace all previous attributes.
    .attributesToProject("hobbies")
    .build();
PageIterable<Person> hobbiesOnlyResult =
    personDynamoDbTable.scan(lastOverwrites);
hobbiesOnlyResult.items().forEach(p ->
    logger.info(p.toString()));

// Logged results.
Person{id=null, firstName='null', lastName='null', age=null, addresses=null,
    phoneNumbers=null, hobbies=[hobby 2, hobby 21]}
Person{id=null, firstName='null', lastName='null', age=null, addresses=null,
    phoneNumbers=null, hobbies=[hobby 1, hobby 11]}
```

Il seguente frammento di codice utilizza innanzitutto il metodo `attributesToProject()`. Questo ordinamento conserva tutti gli altri attributi richiesti.

```
ScanEnhancedRequest attributesPreserved = ScanEnhancedRequest.builder()
    // Use 'attributesToProject()' first so that the method call does not
    replace all other attributes
    // that you want to project.
    .attributesToProject("firstName")
    .addNestedAttributeToProject(
        NestedAttributeName.create("addresses", "work", "street"))
    .addAttributeToProject("hobbies")
    .build();
```

```

PageIterable<Person> allAttributesResult =
    personDynamoDbTable.scan(attributesPreserved);
allAttributesResult.items().forEach(p ->
    logger.info(p.toString()));

// Logged results.
Person{id=null, firstName='first name 2', lastName='null', age=null,
    addresses={work=Address{street='street 21', city='null', state='null',
    zipCode='null'}}}, phoneNumbers=null, hobbies=[hobby 2, hobby 21]}
Person{id=null, firstName='first name 1', lastName='null', age=null,
    addresses={work=Address{street='street 11', city='null', state='null',
    zipCode='null'}}}, phoneNumbers=null, hobbies=[hobby 1, hobby 11]}

```

## Conserva gli oggetti vuoti con `@DynamoDbPreserveEmptyObject`

Se salvi un bean in Amazon DynamoDB con oggetti vuoti e desideri che l'SDK ricrei gli oggetti vuoti al momento del recupero, annota il getter del bean interno con. `@DynamoDbPreserveEmptyObject`

Per illustrare come funziona l'annotazione, l'esempio di codice utilizza i due bean seguenti.

### Fagioli di esempio

La seguente classe di dati contiene due InnerBean campi. Il metodo `getInnerBeanWithoutAnno()`, non è annotato con. `@DynamoDbPreserveEmptyObject` Il `getInnerBeanWithAnno()` metodo è annotato.

```

@DynamoDbBean
public class MyBean {

    private String id;
    private String name;
    private InnerBean innerBeanWithoutAnno;
    private InnerBean innerBeanWithAnno;

    @DynamoDbPartitionKey
    public String getId() { return id; }
    public void setId(String id) { this.id = id; }

    public String getName() { return name; }
    public void setName(String name) { this.name = name; }
}

```

```

    public InnerBean getInnerBeanWithoutAnno() { return innerBeanWithoutAnno; }
    public void setInnerBeanWithoutAnno(InnerBean innerBeanWithoutAnno)
{ this.innerBeanWithoutAnno = innerBeanWithoutAnno; }

    @DynamoDbPreserveEmptyObject
    public InnerBean getInnerBeanWithAnno() { return innerBeanWithAnno; }
    public void setInnerBeanWithAnno(InnerBean innerBeanWithAnno)
{ this.innerBeanWithAnno = innerBeanWithAnno; }

    @Override
    public String toString() {
        return new StringJoiner(", ", MyBean.class.getSimpleName() + "[", "]")
            .add("innerBeanWithoutAnno=" + innerBeanWithoutAnno)
            .add("innerBeanWithAnno=" + innerBeanWithAnno)
            .add("id='" + id + "'")
            .add("name='" + name + "'")
            .toString();
    }
}
}

```

Le istanze della `InnerBean` classe seguente sono campi di `MyBean` e vengono inizializzate come oggetti vuoti nel codice di esempio.

```

@DynamoDbBean
public class InnerBean {

    private String innerBeanField;

    public String getInnerBeanField() {
        return innerBeanField;
    }

    public void setInnerBeanField(String innerBeanField) {
        this.innerBeanField = innerBeanField;
    }

    @Override
    public String toString() {
        return "InnerBean{" +
            "innerBeanField='" + innerBeanField + '\'' +
            '}';
    }
}
}

```

Il seguente esempio di codice salva un MyBean oggetto con bean interni inizializzati in DynamoDB e quindi recupera l'elemento. L'output registrato mostra che non `innerBeanWithoutAnno` è stato inizializzato, ma è stato creato. `innerBeanWithAnno`

```
public MyBean preserveEmptyObjectAnnoUsingGetItemExample(DynamoDbTable<MyBean>
myBeanTable) {
    // Save an item to DynamoDB.
    MyBean bean = new MyBean();
    bean.setId("1");
    bean.setInnerBeanWithoutAnno(new InnerBean()); // Instantiate the inner bean.
    bean.setInnerBeanWithAnno(new InnerBean());   // Instantiate the inner bean.
    myBeanTable.putItem(bean);

    GetItemEnhancedRequest request = GetItemEnhancedRequest.builder()
        .key(Key.builder().partitionValue("1").build())
        .build();
    MyBean myBean = myBeanTable.getItem(request);

    logger.info(myBean.toString());
    // Output 'MyBean[innerBeanWithoutAnno=null,
innerBeanWithAnno=InnerBean{innerBeanField='null'}, id='1', name='null']'.

    return myBean;
}
```

### Schema statico alternativo

È possibile utilizzare la seguente `StaticTableSchema` versione degli schemi delle tabelle al posto delle annotazioni sui bean.

```
public static TableSchema<MyBean> buildStaticSchemas() {

    StaticTableSchema<InnerBean> innerBeanStaticTableSchema =
        StaticTableSchema.builder(InnerBean.class)
            .newItemSupplier(InnerBean::new)
            .addAttribute(String.class, a -> a.name("innerBeanField")
                .getter(InnerBean::getInnerBeanField)
                .setter(InnerBean::setInnerBeanField))
            .build();

    return StaticTableSchema.builder(MyBean.class)
        .newItemSupplier(MyBean::new)
        .addAttribute(String.class, a -> a.name("id"))
```

```

        .getter(MyBean::getId)
        .setter(MyBean::setId)
        .addTag(primaryPartitionKey()))
    .addAttribute(String.class, a -> a.name("name")
        .getter(MyBean::getName)
        .setter(MyBean::setName))
    .addAttribute(EnhancedType.documentOf(InnerBean.class,
        innerBeanStaticTableSchema),
        a -> a.name("innerBean1")
            .getter(MyBean::getInnerBeanWithoutAnno)
            .setter(MyBean::setInnerBeanWithoutAnno))
    .addAttribute(EnhancedType.documentOf(InnerBean.class,
        innerBeanStaticTableSchema,
        b -> b.preserveEmptyObject(true)),
        a -> a.name("innerBean2")
            .getter(MyBean::getInnerBeanWithAnno)
            .setter(MyBean::setInnerBeanWithAnno))
    .build();
}

```

Evita di salvare gli attributi nulli degli oggetti annidati

È possibile ignorare gli attributi nulli degli oggetti nidificati quando si salva un oggetto di classe di dati in DynamoDB applicando l'annotazione. `@DynamoDbIgnoreNulls` Al contrario, gli attributi di primo livello con valori nulli non vengono mai salvati nel database.

Per illustrare come funziona l'annotazione, l'esempio di codice utilizza i due bean seguenti.

Fagioli di esempio

La seguente classe di dati contiene due `InnerBean` campi. Il metodo

`getInnerBeanWithoutAnno()`, non è annotato. Il

`getInnerBeanWithIgnoreNullsAnno()` metodo è annotato con. `@DynamoDbIgnoreNulls`

```

@DynamoDbBean
public class MyBean {

    private String id;
    private String name;
    private InnerBean innerBeanWithoutAnno;
    private InnerBean innerBeanWithIgnoreNullsAnno;

    @DynamoDbPartitionKey

```



```

public String getId() { return id; }
public void setId(String id) { this.id = id; }

public String getName() { return name; }
public void setName(String name) { this.name = name; }

public InnerBean getInnerBeanWithoutAnno() { return innerBeanWithoutAnno; }
public void setInnerBeanWithoutAnno(InnerBean innerBeanWithoutAnno)
{ this.innerBeanWithoutAnno = innerBeanWithoutAnno; }

@DynamoDbIgnoreNulls
public InnerBean getInnerBeanWithIgnoreNullsAnno() { return
innerBeanWithIgnoreNullsAnno; }
public void setInnerBeanWithIgnoreNullsAnno(InnerBean innerBeanWithAnno)
{ this.innerBeanWithIgnoreNullsAnno = innerBeanWithAnno; }

@Override
public String toString() {
    return new StringJoiner(", ", MyBean.class.getSimpleName() + "[", "]")
        .add("innerBeanWithoutAnno=" + innerBeanWithoutAnno)
        .add("innerBeanWithIgnoreNullsAnno=" + innerBeanWithIgnoreNullsAnno)
        .add("id=" + id + "'")
        .add("name=" + name + "'")
        .toString();
}
}

```

Le istanze della seguente InnerBean classe sono campi di MyBean e vengono utilizzate nel seguente codice di esempio.

```

@DynamoDbBean
public class InnerBean {

    private String innerBeanFieldString;
    private Integer innerBeanFieldInteger;

    public String getInnerBeanFieldString() { return innerBeanFieldString; }
    public void setInnerBeanFieldString(String innerBeanFieldString)
{ this.innerBeanFieldString = innerBeanFieldString; }

    public Integer getInnerBeanFieldInteger() { return innerBeanFieldInteger; }
    public void setInnerBeanFieldInteger(Integer innerBeanFieldInteger)
{ this.innerBeanFieldInteger = innerBeanFieldInteger; }
}

```

```

@Override
public String toString() {
    return new StringJoiner(", ", InnerBean.class.getSimpleName() + "[", "]")
        .add("innerBeanFieldString='" + innerBeanFieldString + "'")
        .add("innerBeanFieldInteger=" + innerBeanFieldInteger)
        .toString();
}
}

```

Il seguente esempio di codice crea un `InnerBean` oggetto e imposta solo uno dei suoi due attributi con un valore.

```

public void ignoreNullsAnnoUsingPutItemExample(DynamoDbTable<MyBean> myBeanTable) {
    // Create an InnerBean object and give only one attribute a value.
    InnerBean innerBeanOneAttributeSet = new InnerBean();
    innerBeanOneAttributeSet.setInnerBeanFieldInteger(200);

    // Create a MyBean instance and use the same InnerBean instance both for
attributes.
    MyBean bean = new MyBean();
    bean.setId("1");
    bean.setInnerBeanWithoutAnno(innerBeanOneAttributeSet);
    bean.setInnerBeanWithIgnoreNullsAnno(innerBeanOneAttributeSet);

    Map<String, AttributeValue> itemMap = myBeanTable.tableSchema().itemToMap(bean,
true);
    logger.info(itemMap.toString());
    // Log the map that is sent to the database.
    //
    {innerBeanWithIgnoreNullsAnno=AttributeValue(M={innerBeanFieldInteger=AttributeValue(N=200)}),
id=AttributeValue(S=1),
innerBeanWithoutAnno=AttributeValue(M={innerBeanFieldInteger=AttributeValue(N=200),
innerBeanFieldString=AttributeValue(NUL=true)}})}

    // Save the MyBean object to the table.
    myBeanTable.putItem(bean);
}

```

Per visualizzare i dati di basso livello inviati a DynamoDB, il codice registra la mappa degli attributi prima di salvare l'oggetto. `MyBean`

L'output registrato mostra che restituisce un attributo, `innerBeanWithIgnoreNullsAnno`

```
innerBeanWithIgnoreNullsAnno=AttributeValue(M={innerBeanFieldInteger=AttributeValue(N=200)}))
```

L'`innerBeanWithoutAnno` restituisce due attributi. Un attributo ha un valore di 200 e l'altro è un attributo con valore nullo.

```
innerBeanWithoutAnno=AttributeValue(M={innerBeanFieldInteger=AttributeValue(N=200),
innerBeanFieldString=AttributeValue(NULL=true)})
```

## Rappresentazione JSON della mappa degli attributi

La seguente rappresentazione JSON semplifica la visualizzazione dei dati salvati in DynamoDB.

```
{
  "id": {
    "S": "1"
  },
  "innerBeanWithIgnoreNullsAnno": {
    "M": {
      "innerBeanFieldInteger": {
        "N": "200"
      }
    }
  },
  "innerBeanWithoutAnno": {
    "M": {
      "innerBeanFieldInteger": {
        "N": "200"
      },
      "innerBeanFieldString": {
        "NULL": true
      }
    }
  }
}
```

## Schema statico alternativo

È possibile utilizzare la seguente `StaticTableSchema` versione degli schemi delle tabelle al posto delle annotazioni delle classi di dati.

```
public static TableSchema<MyBean> buildStaticSchemas() {
```

```

StaticTableSchema<InnerBean> innerBeanStaticTableSchema =
    StaticTableSchema.builder(InnerBean.class)
        .newItemSupplier(InnerBean::new)
        .addAttribute(String.class, a -> a.name("innerBeanFieldString")
            .getter(InnerBean::getInnerBeanFieldString)
            .setter(InnerBean::setInnerBeanFieldString))
        .addAttribute(Integer.class, a -> a.name("innerBeanFieldInteger")
            .getter(InnerBean::getInnerBeanFieldInteger)
            .setter(InnerBean::setInnerBeanFieldInteger))
        .build();

return StaticTableSchema.builder(MyBean.class)
    .newItemSupplier(MyBean::new)
    .addAttribute(String.class, a -> a.name("id")
        .getter(MyBean::getId)
        .setter(MyBean::setId)
        .addTag(primaryPartitionKey()))
    .addAttribute(String.class, a -> a.name("name")
        .getter(MyBean::getName)
        .setter(MyBean::setName))
    .addAttribute(EnhancedType.documentOf(InnerBean.class,
        innerBeanStaticTableSchema),
        a -> a.name("innerBeanWithoutAnno")
            .getter(MyBean::getInnerBeanWithoutAnno)
            .setter(MyBean::setInnerBeanWithoutAnno))
    .addAttribute(EnhancedType.documentOf(InnerBean.class,
        innerBeanStaticTableSchema,
        b -> b.ignoreNulls(true)),
        a -> a.name("innerBeanWithIgnoreNullsAnno")
            .getter(MyBean::getInnerBeanWithIgnoreNullsAnno)
            .setter(MyBean::setInnerBeanWithIgnoreNullsAnno))
    .build();
}

```

## Lavora con documenti JSON con l'API Enhanced Document per DynamoDB

L'[Enhanced Document API](#) for AWS SDK for Java 2.x è progettata per funzionare con dati orientati ai documenti che non hanno uno schema fisso. Tuttavia, consente anche di utilizzare classi personalizzate per mappare singoli attributi.

L'Enhanced Document API è il successore dell'[API Document](#) della AWS SDK for Java versione 1.x.

### Indice

- [Inizia a usare l'Enhanced Document API](#)
  - [Crea un e un DocumentTableSchemaDynamoDbTable](#)
- [Crea documenti avanzati](#)
  - [Crea da una stringa JSON](#)
  - [Costruisci a partire da singoli elementi](#)
- [Esegui operazioni CRUD](#)
- [Accedi agli attributi avanzati del documento come oggetti personalizzati](#)
- [Usa e EnhancedDocument senza DynamoDB](#)

Inizia a usare l'Enhanced Document API

L'API Enhanced Document richiede le stesse [dipendenze](#) necessarie per l'API DynamoDB Enhanced Client. Richiede anche un [DynamoDbEnhancedClientistanza](#), come mostrato all'inizio di questo argomento.

Poiché l'Enhanced Document API è stata rilasciata con la versione 2.20.3 di AWS SDK for Java 2.x, è necessaria quella versione o superiore.

### Crea un e un **DocumentTableSchemaDynamoDbTable**

Per richiamare comandi su una tabella DynamoDB utilizzando l'API Enhanced Document, associa la tabella a un oggetto risorsa < > lato [DynamoDbTableclient EnhancedDocument](#).

Il `table()` metodo del client avanzato crea un `DynamoDbTable<EnhancedDocument>` istanza e richiede parametri per il nome della tabella DynamoDB e un `DocumentTableSchema`

Il builder for a [DocumentTableSchema](#) richiede una chiave di indice principale e uno o più provider di convertitori di attributi. Il `AttributeConverterProvider.defaultProvider()` metodo fornisce convertitori per i tipi [predefiniti](#). Dovrebbe essere specificato anche se si fornisce un provider di convertitori di attributi personalizzato. È possibile aggiungere una chiave di indice secondaria opzionale al generatore.

Il seguente frammento di codice mostra il codice che genera la rappresentazione lato client di una tabella DynamoDB che memorizza oggetti senza schema. `person EnhancedDocument`

```
DynamoDbTable<EnhancedDocument> documentDynamoDbTable =
    enhancedClient.table("person",
        TableSchema.documentSchemaBuilder())
```

```
        // Specify the primary key attributes.

        .addIndexPartitionKey(TableMetadata.primaryIndexName(),"id", AttributeValueType.S)
            .addIndexSortKey(TableMetadata.primaryIndexName(),
                "lastName", AttributeValueType.S)
            // Specify attribute converter providers. Minimally add the
            default one.

        .attributeConverterProviders(AttributeConverterProvider.defaultProvider())
            .build());

// Call documentTable.createTable() if "person" does not exist in DynamoDB.
// createTable() should be called only one time.
```

Di seguito viene illustrata la rappresentazione JSON di un oggetto utilizzata in questa sezione.  
person

### Oggetto JSON **person**

```
{
  "id": 1,
  "firstName": "Richard",
  "lastName": "Roe",
  "age": 25,
  "addresses":
  {
    "home": {
      "zipCode": "00000",
      "city": "Any Town",
      "state": "FL",
      "street": "123 Any Street"
    },
    "work": {
      "zipCode": "00001",
      "city": "Anywhere",
      "state": "FL",
      "street": "100 Main Street"
    }
  },
  "hobbies": [
    "Hobby 1",
    "Hobby 2"
  ],
}
```

```
"phoneNumbers": [  
  {  
    "type": "Home",  
    "number": "555-0100"  
  },  
  {  
    "type": "Work",  
    "number": "555-0119"  
  }  
]
```

## Crea documenti avanzati

An [EnhancedDocument](#) rappresenta un oggetto di tipo documento con una struttura complessa con attributi annidati. An EnhancedDocument richiede attributi di primo livello che corrispondano agli attributi della chiave primaria specificati per. DocumentTableSchema Il contenuto rimanente è arbitrario e può essere costituito da attributi di primo livello e anche da attributi profondamente annidati.

Si crea un'EnhancedDocumentistanza utilizzando un generatore che offre diversi modi per aggiungere elementi.

### Crea da una stringa JSON

Con una stringa JSON, puoi creare una chiamata EnhancedDocument in un unico metodo. Il seguente frammento crea un EnhancedDocument da una stringa JSON restituita dal metodo helper. [jsonPerson\(\)](#) [Il jsonPerson\(\) metodo restituisce la versione in stringa JSON dell'oggetto persona mostrato in precedenza.](#)

```
EnhancedDocument document =  
    EnhancedDocument.builder()  
        .json( jsonPerson() )  
        .build();
```

### Costruisci a partire da singoli elementi

In alternativa, puoi creare un'EnhancedDocumentistanza a partire da singoli componenti utilizzando i metodi type-safe del builder.

L'esempio seguente crea un documento person avanzato simile al documento avanzato creato a partire dalla stringa JSON dell'esempio precedente.

/\* Define the shape of an address map whose JSON representation looks like the following.

Use 'addressMapEnhancedType' in the following EnhancedDocument.builder() to simplify the code.

```
"home": {
  "zipCode": "00000",
  "city": "Any Town",
  "state": "FL",
  "street": "123 Any Street"
}*/
```

```
EnhancedType<Map<String, String>> addressMapEnhancedType =
    EnhancedType.mapOf(EnhancedType.of(String.class),
EnhancedType.of(String.class));
```

// Use the builder's typesafe methods to add elements to the enhanced document.

```
EnhancedDocument personDocument = EnhancedDocument.builder()
    .putNumber("id", 50)
    .putString("firstName", "Shirley")
    .putString("lastName", "Rodriguez")
    .putNumber("age", 53)
    .putNull("nullAttribute")
    .putJson("phoneNumbers", phoneNumbersJSONString())
```

/\* Add the map of addresses whose JSON representation looks like the following.

```
{
  "home": {
    "zipCode": "00000",
    "city": "Any Town",
    "state": "FL",
    "street": "123 Any Street"
  }
} */
    .putMap("addresses", getAddresses(), EnhancedType.of(String.class),
addressMapEnhancedType)
    .putList("hobbies", List.of("Theater", "Golf"),
EnhancedType.of(String.class))
    .build();
```

## Metodi di supporto

```
private static String phoneNumbersJSONString() {
```



```

        return " [" +
            " {" +
            "   \"type\": \"Home\", " +
            "   \"number\": \"555-0140\"" +
            " }, " +
            " {" +
            "   \"type\": \"Work\", " +
            "   \"number\": \"555-0155\"" +
            " }" +
            " ]";
    }

    private static Map<String, Map<String, String>> getAddresses() {
        return Map.of(
            "home", Map.of(
                "zipCode", "00002",
                "city", "Any Town",
                "state", "ME",
                "street", "123 Any Street"));
    }
}

```

## Esegui operazioni CRUD

Dopo aver definito un'EnhancedDocumentistanza, è possibile salvarla in una tabella DynamoDB. Il seguente frammento di codice utilizza [PersonDocument creato](#) da singoli elementi.

```
documentDynamoDbTable.putItem(personDocument);
```

Dopo aver letto un'istanza di documento avanzata da DynamoDB, è possibile estrarre i valori dei singoli attributi utilizzando i getter, come mostrato nel seguente frammento di codice, che accedono ai dati salvati da `personDocument`. In alternativa, è possibile estrarre il contenuto completo in una stringa JSON, come mostrato nell'ultima parte del codice di esempio.

```

// Read the item.
EnhancedDocument personDocFromDb =
documentDynamoDbTable.getItem(Key.builder().partitionValue(50).build());

// Access top-level attributes.
logger.info("Name: {} {}", personDocFromDb.getString("firstName"),
personDocFromDb.getString("lastName"));
// Name: Shirley Rodriguez

```

```

    // Typesafe access of a deeply nested attribute. The addressMapEnhancedType
    shown previously defines the shape of an addresses map.
    Map<String, Map<String, String>> addresses =
    personDocFromDb.getMap("addresses", EnhancedType.of(String.class),
    addressMapEnhancedType);
    addresses.keySet().forEach(k -> logger.info(addresses.get(k).toString()));
    // {zipCode=00002, city=Any Town, street=123 Any Street, state=ME}

    // Alternatively, work with AttributeValue types checking along the way for
    deeply nested attributes.
    Map<String, AttributeValue> addressesMap =
    personDocFromDb.getMapOfUnknownType("addresses");
    addressesMap.keySet().forEach((String k) -> {
        logger.info("Looking at data for [{}] address", k);
        // Looking at data for [home] address
        AttributeValue value = addressesMap.get(k);
        AttributeValue cityValue = value.m().get("city");
        if (cityValue != null) {
            logger.info(cityValue.s());
            // Any Town
        }
    });

    List<AttributeValue> phoneNumbers =
    personDocFromDb.getListOfUnknownType("phoneNumbers");
    phoneNumbers.forEach((AttributeValue av) -> {
        if (av.hasM()) {
            AttributeValue type = av.m().get("type");
            if (type.s() != null) {
                logger.info("Type of phone: {}", type.s());
                // Type of phone: Home
                // Type of phone: Work
            }
        }
    });

    String jsonPerson = personDocFromDb.toJson();
    logger.info(jsonPerson);
    // {"firstName":"Shirley","lastName":"Rodriguez","addresses":
{"home":{"zipCode":"00002","city":"Any Town","street":"123 Any
Street","state":"ME"}}, "hobbies":["Theater","Golf"],

```

```
//      "id":50,"nullAttribute":null,"age":53,"phoneNumbers":
[{"number":"555-0140","type":"Home"},{"number":"555-0155","type":"Work"}]}
```

EnhancedDocument le istanze possono essere utilizzate con qualsiasi metodo [DynamoDbTable](#) o al posto delle classi [DynamoDbEnhancedClient](#) di dati mappate.

Accedi agli attributi avanzati del documento come oggetti personalizzati

Oltre a fornire un'API per leggere e scrivere attributi con strutture senza schema, l'API Enhanced Document consente di convertire gli attributi da e verso istanze di classi personalizzate.

L'API Enhanced Document utilizza `AttributeConverterProvider` i `AttributeConverter` caratteri e i caratteri mostrati nella sezione di [conversione degli attributi di controllo](#) come parte dell'API DynamoDB Enhanced Client.

Nell'esempio seguente, utilizziamo a `CustomAttributeConverterProvider` con la sua `AddressConverter` classe annidata per convertire gli oggetti. `Address`

Questo esempio mostra che è possibile combinare dati provenienti da classi e anche dati provenienti da strutture create secondo necessità. Questo esempio mostra anche che le classi personalizzate possono essere utilizzate a qualsiasi livello di una struttura annidata. Gli `Address` oggetti in questo esempio sono valori utilizzati in una mappa.

```
public static void attributeToAddressClassMappingExample(DynamoDbEnhancedClient
enhancedClient, DynamoDbClient standardClient) {
    String tableName = "customer";

    // Define the DynamoDbTable for an enhanced document.
    // The schema builder provides methods for attribute converter providers and
keys.
    DynamoDbTable<EnhancedDocument> documentDynamoDbTable =
enhancedClient.table(tableName,
        DocumentTableSchema.builder()
            // Add the CustomAttributeConverterProvider along with the
default when you build the table schema.
            .attributeConverterProviders(
                List.of(
                    new CustomAttributeConverterProvider(),
                    AttributeConverterProvider.defaultProvider()))
            .addIndexPartitionKey(TableMetadata.primaryIndexName(), "id",
AttributeValueType.N)
            .addIndexSortKey(TableMetadata.primaryIndexName(), "lastName",
AttributeValueType.S)
```

```

        .build());
// Create the DynamoDB table if needed.
documentDynamoDbTable.createTable();
waitForTableCreation(tableName, standardClient);

// The getAddressessForCustomMappingExample() helper method that provides
'addresses' shows the use of a custom Address class
// rather than using a Map<String, Map<String, String> to hold the address
data.
Map<String, Address> addresses = getAddressessForCustomMappingExample();

// Build an EnhancedDocument instance to save an item with a mix of structures
defined as needed and static classes.
EnhancedDocument personDocument = EnhancedDocument.builder()
    .putNumber("id", 50)
    .putString("firstName", "Shirley")
    .putString("lastName", "Rodriguez")
    .putNumber("age", 53)
    .putNull("nullAttribute")
    .putJson("phoneNumbers", phoneNumbersJSONString())
    // Note the use of 'EnhancedType.of(Address.class)' instead of the more
generic
    // 'EnhancedType.mapOf(EnhancedType.of(String.class),
EnhancedType.of(String.class))' that was used in a previous example.
    .putMap("addresses", addresses, EnhancedType.of(String.class),
EnhancedType.of(Address.class))
    .putList("hobbies", List.of("Hobby 1", "Hobby 2"),
EnhancedType.of(String.class))
    .build();
// Save the item to DynamoDB.
documentDynamoDbTable.putItem(personDocument);

// Retrieve the item just saved.
EnhancedDocument srPerson =
documentDynamoDbTable.getItem(Key.builder().partitionValue(50).sortValue("Rodriguez").build())

// Access the addresses attribute.
Map<String, Address> srAddresses = srPerson.get("addresses",
    EnhancedType.mapOf(EnhancedType.of(String.class),
EnhancedType.of(Address.class)));

srAddresses.keySet().forEach(k -> logger.info(addresses.get(k).toString()));

```

```

        documentDynamoDbTable.deleteTable();

// The content logged to the console shows that the saved maps were converted to
// Address instances.
Address{street='123 Main Street', city='Any Town', state='NC', zipCode='00000'}
Address{street='100 Any Street', city='Any Town', state='NC', zipCode='00000'}

```

## CustomAttributeConverterProvider codice

```

public class CustomAttributeConverterProvider implements AttributeConverterProvider {

    private final Map<EnhancedType<?>, AttributeConverter<?>> converterCache =
ImmutableMap.of(
        // 1. Add AddressConverter to the internal cache.
        EnhancedType.of(Address.class), new AddressConverter());

    public static CustomAttributeConverterProvider create() {
        return new CustomAttributeConverterProvider();
    }

    // 2. The enhanced client queries the provider for attribute converters if it
    // encounters a type that it does not know how to convert.
    @SuppressWarnings("unchecked")
    @Override
    public <T> AttributeConverter<T> converterFor(EnhancedType<T> enhancedType) {
        return (AttributeConverter<T>) converterCache.get(enhancedType);
    }

    // 3. Custom attribute converter
    private class AddressConverter implements AttributeConverter<Address> {
        // 4. Transform an Address object into a DynamoDB map.
        @Override
        public AttributeValue transformFrom(Address address) {

            Map<String, AttributeValue> attributeValueMap = Map.of(
                "street", AttributeValue.fromS(address.getStreet()),
                "city", AttributeValue.fromS(address.getCity()),
                "state", AttributeValue.fromS(address.getState()),
                "zipCode", AttributeValue.fromS(address.getZipCode()));

            return AttributeValue.fromM(attributeValueMap);
        }
    }
}

```

```
// 5. Transform the DynamoDB map attribute to an Address object.
@Override
public Address transformTo(AttributeValue attributeValue) {
    Map<String, AttributeValue> m = attributeValue.m();
    Address address = new Address();
    address.setStreet(m.get("street").s());
    address.setCity(m.get("city").s());
    address.setState(m.get("state").s());
    address.setZipCode(m.get("zipCode").s());

    return address;
}

@Override
public EnhancedType<Address> type() {
    return EnhancedType.of(Address.class);
}

@Override
public AttributeValueType attributeValueType() {
    return AttributeValueType.M;
}
}
```

## Classe Address

```
public class Address {
    private String street;
    private String city;
    private String state;
    private String zipCode;

    public Address() {
    }

    public String getStreet() {
    return this.street;
    }

    public String getCity() {
    return this.city;
    }
}
```

```
public String getState() {
    return this.state;
}

public String getZipCode() {
    return this.zipCode;
}

public void setStreet(String street) {
    this.street = street;
}

public void setCity(String city) {
    this.city = city;
}

public void setState(String state) {
    this.state = state;
}

public void setZipCode(String zipCode) {
    this.zipCode = zipCode;
}
}
```

Metodo di supporto che fornisce indirizzi

Il seguente metodo di supporto fornisce la mappa che utilizza istanze personalizzate per i valori anziché Address istanze generiche `Map<String, String>` per i valori.

```
private static Map<String, Address> getAddressesForCustomMappingExample() {
    Address homeAddress = new Address();
    homeAddress.setStreet("100 Any Street");
    homeAddress.setCity("Any Town");
    homeAddress.setState("NC");
    homeAddress.setZipCode("00000");

    Address workAddress = new Address();
    workAddress.setStreet("123 Main Street");
    workAddress.setCity("Any Town");
    workAddress.setState("NC");
    workAddress.setZipCode("00000");
}
```

```

    return Map.of("home", homeAddress,
                 "work", workAddress);
}

```

## Usa e **EnhancedDocument** senza DynamoDB

Sebbene di solito si utilizzi un'istanza di un `EnhancedDocument` per leggere e scrivere elementi DynamoDB di tipo documento, può essere utilizzata anche indipendentemente da DynamoDB.

È possibile utilizzarli `EnhancedDocuments` per la loro capacità di convertire tra stringhe JSON o oggetti personalizzati in mappe di basso livello come mostrato nell'esempio seguente.

### AttributeValues

```

public static void conversionWithoutDynamoDbExample() {
    Address address = new Address();
    address.setCity("my city");
    address.setState("my state");
    address.setStreet("my street");
    address.setZipCode("00000");

    // Build an EnhancedDocument instance for its conversion functionality alone.
    EnhancedDocument addressEnhancedDoc = EnhancedDocument.builder()
        // Important: You must specify attribute converter providers when you
        // build an EnhancedDocument instance not used with a DynamoDB table.
        .attributeConverterProviders(new CustomAttributeConverterProvider(),
        DefaultAttributeConverterProvider.create())
        .put("addressDoc", address, Address.class)
        .build();

    // Convert address to a low-level item representation.
    final Map<String, AttributeValue> addressAsAttributeMap =
addressEnhancedDoc.getMapOfUnknownType("addressDoc");
    logger.info("addressAsAttributeMap: {}", addressAsAttributeMap.toString());

    // Convert address to a JSON string.
    String addressAsJsonString = addressEnhancedDoc.getJson("addressDoc");
    logger.info("addressAsJsonString: {}", addressAsJsonString);
    // Convert addressEnhancedDoc back to an Address instance.
    Address addressConverted = addressEnhancedDoc.get("addressDoc",
Address.class);
    logger.info("addressConverted: {}", addressConverted.toString());
}

```



```
/* Console output:
    addressAsAttributeMap: {zipCode=AttributeValue(S=00000),
state=AttributeValue(S=my state), street=AttributeValue(S=my street),
city=AttributeValue(S=my city)}
    addressAsJsonString: {"zipCode":"00000","state":"my state","street":"my
street","city":"my city"}
    addressConverted: Address{street='my street', city='my city', state='my
state', zipCode='00000'}
*/
```

### Note

Quando utilizzi un documento avanzato indipendente da una tabella DynamoDB, assicurati di impostare in modo esplicito i provider di conversione degli attributi sul builder.

Al contrario, lo schema della tabella dei documenti fornisce i provider di conversione quando un documento avanzato viene utilizzato con una tabella DynamoDB.

## Usa le estensioni

L'API DynamoDB Enhanced Client supporta le estensioni dei plugin che forniscono funzionalità che vanno oltre le operazioni di mappatura. Le estensioni hanno due metodi hook e. `beforeWrite()` `afterRead()` `beforeWrite()` modifica un'operazione di scrittura prima che venga eseguita e il `afterRead()` metodo modifica i risultati di un'operazione di lettura dopo l'operazione. Poiché alcune operazioni (come l'aggiornamento degli elementi) eseguono sia un'operazione di scrittura che una successiva lettura, vengono chiamati entrambi i metodi hook.

Le estensioni vengono caricate nell'ordine in cui sono specificate nel generatore di client avanzato. L'ordine di caricamento può essere importante perché un'estensione può agire su valori che sono stati trasformati da un'estensione precedente.

L'API client avanzata include un set di estensioni di plug-in che si trovano nel [extensions](#) pacchetto. Per impostazione predefinita, il client avanzato carica [VersionedRecordExtension](#) e il [AtomicCounterExtension](#). È possibile sovrascrivere il comportamento predefinito con Enhance Client Builder e caricare qualsiasi estensione. Puoi anche specificarne nessuna se non desideri le estensioni predefinite.

Se carichi le tue estensioni, il client avanzato non carica alcuna estensione predefinita. Se desideri il comportamento fornito da una delle estensioni predefinite, devi aggiungerla esplicitamente all'elenco delle estensioni.

Nell'esempio seguente, un'estensione personalizzata denominata `verifyChecksumExtension` viene caricata dopo `VersionedRecordExtension`, che di solito viene caricata automaticamente per impostazione predefinita. In questo esempio non `AtomicCounterExtension` viene caricato.

```
DynamoDbEnhancedClientExtension versionedRecordExtension =
    VersionedRecordExtension.builder().build();

DynamoDbEnhancedClient enhancedClient =
    DynamoDbEnhancedClient.builder()
        .dynamoDbClient(dynamoDbClient)
        .extensions(versionedRecordExtension,
verifyChecksumExtension)
        .build();
```

## VersionedRecordExtension

`VersionedRecordExtension` viene caricato per impostazione predefinita e incrementa e tiene traccia del numero di versione di un elemento man mano che gli elementi vengono scritti nel database. A ogni scrittura verrà aggiunta una condizione che causa l'esito negativo della scrittura se il numero di versione dell'elemento persistente effettivo non corrisponde al valore letto l'ultima volta dall'applicazione. Questo comportamento fornisce in modo efficace un blocco ottimistico per gli aggiornamenti degli elementi. Se un altro processo aggiorna un elemento tra il momento in cui il primo processo ha letto l'elemento e lo sta scrivendo, la scrittura avrà esito negativo.

Per specificare l'attributo da utilizzare per tenere traccia del numero di versione dell'articolo, contrassegnate un attributo numerico nello schema della tabella.

Il frammento seguente specifica che l'`versionattributo` deve contenere il numero di versione dell'articolo.

```
@DynamoDbVersionAttribute
public Integer getVersion() {...};
public void setVersion(Integer version) {...};
```

L'approccio equivalente allo schema di tabella statica è illustrato nel frammento seguente.

```

.addAttribute(Integer.class, a -> a.name("version")
                .getter(Customer::getVersion)
                .setter(Customer::setVersion)
                // Apply the 'version' tag to the attribute.

.tags(VersionedRecordExtension.AttributeTags.versionAttribute())

```

## AtomicCounterExtension

`AtomicCounterExtension` viene caricato per impostazione predefinita e incrementa un attributo numerico con tag ogni volta che un record viene scritto nel database. È possibile specificare valori iniziali e incrementali. Se non viene specificato alcun valore, il valore iniziale viene impostato su 0 e il valore dell'attributo aumenta di 1.

Per specificare quale attributo è un contatore, contrassegnate un attributo di tipo Long nello schema della tabella.

Il frammento seguente mostra l'uso dei valori di inizio e incremento predefiniti per l'attributo `counter`

```

@DynamoDbAtomicCounter
public Long getCounter() {...};
public void setCounter(Long counter) {...};

```

L'approccio allo schema di tabella statica è illustrato nel frammento seguente. L'estensione del contatore atomico utilizza un valore iniziale di 10 e incrementa il valore di 5 ogni volta che viene scritto il record.

```

.addAttribute(Integer.class, a -> a.name("counter")
                .getter(Customer::getCounter)
                .setter(Customer::setCounter)
                // Apply the 'atomicCounter' tag to the
attribute with start and increment values.
                .tags(StaticAttributeTags.atomicCounter(10L,
5L))

```

## AutoGeneratedTimestampRecordExtension

Aggiorna `AutoGeneratedTimestampRecordExtension` automaticamente gli attributi di tipo contrassegnati [Instant](#) con un timestamp corrente ogni volta che l'elemento viene scritto correttamente nel database.

Questa estensione non viene caricata per impostazione predefinita. Pertanto, è necessario specificarla come estensione personalizzata quando si crea il client avanzato, come illustrato nel primo esempio di questo argomento.

Per specificare l'attributo da aggiornare con il timestamp corrente, contrassegnate l'Instant attributo nello schema della tabella.

L'lastUpdate attributo è l'obiettivo del comportamento delle estensioni nel seguente frammento. Nota il requisito secondo cui l'attributo deve essere di Instant tipo.

```
@DynamoDbAutoGeneratedTimestampAttribute
public Instant getLastUpdate() {...}
public void setLastUpdate(Instant lastUpdate) {...}
```

L'approccio equivalente allo schema di tabella statica è illustrato nel frammento seguente.

```
.addAttribute(Instant.class, a -> a.name("lastUpdate")
                .getter(Customer::getLastUpdate)
                .setter(Customer::setLastUpdate)
                // Applying the 'autoGeneratedTimestamp' tag to
the attribute.

.tags(AutoGeneratedTimestampRecordExtension.AttributeTags.autoGeneratedTimestampAttribute())
```

## Estensioni personalizzate

La seguente classe di estensione personalizzata mostra un beforeWrite() metodo che utilizza un'espressione di aggiornamento. Dopo la riga di commento 2, creiamo un SetAction attributo per impostare l'registrationDate attributo se l'elemento nel database non ha già un registrationDate attributo. Ogni volta che un Customer oggetto viene aggiornato, l'estensione si assicura che a registrationDate sia impostato.

```
public final class CustomExtension implements DynamoDbEnhancedClientExtension {

    // 1. In a custom extension, use an UpdateExpression to define what action to take
before
    //    an item is updated.
    @Override
    public WriteModification beforeWrite(DynamoDbExtensionContext.BeforeWrite context)
    {
        if ( context.operationContext().tableName().equals("Customer")
```

```

        && context.operationName().equals(OperationName.UPDATE_ITEM)) {
            return WriteModification.builder()
                .updateExpression(createUpdateExpression())
                .build();
        }
        return WriteModification.builder().build(); // Return an "empty"
WriteModification instance if the extension should not be applied.
                                                    // In this case, if the code is
not updating an item on the Customer table.
    }

    private static UpdateExpression createUpdateExpression() {

        // 2. Use a SetAction, a subclass of UpdateAction, to provide the values in the
update.
        SetAction setAction =
            SetAction.builder()
                .path("registrationDate")
                .value("if_not_exists(registrationDate, :regValue)")
                .putExpressionValue(":regValue",
AttributeValue.fromS(Instant.now().toString()))
                .build();
        // 3. Build the UpdateExpression with one or more UpdateAction.
        return UpdateExpression.builder()
            .addAction(setAction)
            .build();
    }
}

```

## Usa l'API DynamoDB Enhanced Client in modo asincrono

Se l'applicazione richiede chiamate asincrone non bloccanti a DynamoDB, è possibile utilizzare il [DynamoDbEnhancedAsyncClient](#). È simile all'implementazione sincrona ma con le seguenti differenze chiave:

1. Quando si crea il `DynamoDbEnhancedAsyncClient`, è necessario fornire la versione asincrona del client standard `DynamoDbAsyncClient`, come mostrato nel frammento seguente.

```

DynamoDbEnhancedAsyncClient enhancedClient =
    DynamoDbEnhancedAsyncClient.builder()
        .dynamoDbClient(dynamoDbAsyncClient)
        .build();

```

2. I metodi che restituiscono un singolo oggetto di dati restituiscono un `CompletableFuture` risultato anziché solo il risultato. L'applicazione può quindi eseguire altre operazioni senza dover bloccare il risultato. Il frammento seguente mostra il metodo `getItem()` asincrono.

```
CompletableFuture<Customer> result = customerDynamoDbTable.getItem(customer);
// Perform other work here.
return result.join(); // Now block and wait for the result.
```

3. I metodi che restituiscono elenchi di risultati impaginati restituiscono un valore [SdkPublisher](#) anziché un valore restituito dalla modalità sincrona per [SdkIterable](#) negli stessi metodi. `DynamoDbEnhanceClient` L'applicazione può quindi sottoscrivere un gestore a tale editore per gestire i risultati in modo asincrono senza dover bloccare.

```
PagePublisher<Customer> results = customerDynamoDbTable.query(r ->
    r.queryConditional(keyEqualTo(k -> k.partitionValue("Smith"))));
results.subscribe(myCustomerResultsProcessor);
// Perform other work and let the processor handle the results asynchronously.
```

Per un esempio più completo di utilizzo di `SdkPublisher` API, consultate l'[esempio](#) nella sezione che illustra il metodo `scan()` asincrono di questa guida.

## Annotazioni di classi di dati

La tabella seguente elenca le annotazioni che possono essere utilizzate sulle classi di dati e fornisce collegamenti a informazioni ed esempi contenuti in questa guida. La tabella è ordinata in ordine alfabetico crescente in base al nome dell'annotazione.

Annotazioni delle classi di dati utilizzate in questa guida

Nome dell'annotazione	L'annotazione si applica a 1	Cosa fa	Dove viene mostrato in questa guida
<code>DynamoDbAtomicCounter</code>	attributo <sup>2</sup>	Incrementa un attributo numerico con tag ogni volta che un record viene scritto nel database.	<a href="#">Introduzione e discussione.</a>

Nome dell'annotazione	L'annotazione si applica a 1	Cosa fa	Dove viene mostrato in questa guida
DynamoDbAttribute	attributo	Definisce o rinomina una proprietà del bean mappata a un attributo di tabella DynamoDB.	<ul style="list-style-type: none"> <li>• <a href="#">Discussione iniziale.</a></li> <li>• <a href="#">Sezione introduttiva: vedi Nota.</a></li> <li>• <a href="#">In MovieActor classe, esempi del metodo In Query.</a></li> </ul>
DynamoDbAttributeGeneratedTimestampAttribute	attributo	Aggiorna un attributo taggato con un timestamp corrente ogni volta che l'elemento viene scritto correttamente nel database	<a href="#">Introduzione e discussione.</a>
DynamoDbBean	classe	Contrassegna una classe di dati come mappabile su uno schema tabellare.	Primo utilizzo sulla <a href="#">classe Customer nella sezione Guida introduttiva</a> . In tutta la guida sono presenti diversi utilizzi.
DynamoDbConvertedBy	attributo	Associa un attributo personalizzato all'Attribute Converter attributo annotato.	<a href="#">Discussione iniziale ed esempio.</a>

Nome dell'annotazione	L'annotazione si applica a 1	Cosa fa	Dove viene mostrato in questa guida
DynamoDbFlatten	attributo	Appiattisce tutti gli attributi di una classe di dati DynamoDB separata e li aggiunge come attributi di primo livello al record che viene letto e scritto nel database.	<ul style="list-style-type: none"> <li>• <a href="#">Discussione iniziale.</a></li> <li>• <a href="#">Implicazioni per altro codice.</a></li> </ul>
DynamoDbIgnore	attributo	Fa sì che l'attributo rimanga non mappato.	<ul style="list-style-type: none"> <li>• <a href="#">Discussione iniziale.</a></li> <li>• <a href="#">Uso in ProductCatalog classe.</a></li> </ul>
DynamoDbIgnoreNulls	attributo	Impedisce il salvataggio degli attributi nulli degli oggetti annidati. DynamoDb	<a href="#">Discussioni ed esempi.</a>
DynamoDbImmutable	classe	Contrassegna una classe di dati immutabile come mappabile su uno schema tabellare.	<ul style="list-style-type: none"> <li>• <a href="#">Introduzione all'annotazione.</a></li> <li>• <a href="#">Uso in ProductCatalog classe.</a></li> <li>• <a href="#">Da usare con Lombok.</a></li> </ul>
DynamoDbPartitionKey	attributo	Contrassegna un attributo come chiave di partizione primaria (chiave hash) della tabella. DynamoDb	<ul style="list-style-type: none"> <li>• <a href="#">Utilizzo iniziale della classe Customer nella sezione Guida introduttiva.</a></li> <li>• <a href="#">Con Lombok.</a></li> </ul>



Nome dell'annotazione	L'annotazione si applica a 1	Cosa fa	Dove viene mostrato in questa guida
DynamoDbP reserveEmptyObject	attributo	Specifica che se non sono presenti dati per l'oggetto mappato all'attributo annotato, l'oggetto deve essere inizializzato con tutti i campi nulli.	<a href="#">Discussione ed esempi.</a>
DynamoDbS econdaryPartitionKey	attributo	Contrassegna un attributo come chiave di partizione per un indice secondario globale.	<ul style="list-style-type: none"> <li>• <a href="#">Utilizza negli indici secondari ed esempi.</a></li> <li>• <a href="#">Negli esempi del metodo Query.</a></li> <li>• <a href="#">Nell'esempio di Lombok</a></li> <li>• <a href="#">Con classi immutabili.</a></li> </ul>
DynamoDbS econdarySortKey	attributo	Contrassegna un attributo come chiave di ordinamento opzionale per un indice secondario globale o locale.	<ul style="list-style-type: none"> <li>• <a href="#">Utilizza negli indici secondari e negli esempi.</a></li> <li>• <a href="#">Negli esempi del metodo Query.</a></li> <li>• <a href="#">Nell'esempio di Lombok.</a></li> <li>• <a href="#">Con classi immutabili.</a></li> </ul>

Nome dell'annotazione	L'annotazione si applica a <sup>1</sup>	Cosa fa	Dove viene mostrato in questa guida
DynamoDbSortKey	attributo	Contrassegna un attributo come chiave di ordinamento primaria opzionale (chiave di intervallo).	<ul style="list-style-type: none"> <li>• <a href="#">Sezione introduttiva sulla classe Customer.</a></li> <li>• <a href="#">Con classi immutabili.</a></li> <li>• <a href="#">Nell'esempio di Lombok.</a></li> <li>• <a href="#">Negli esempi del metodo Query.</a></li> </ul>
DynamoDbUpdateBehavior	attributo	Specifica il comportamento quando questo attributo viene aggiornato come parte di un'operazione di 'aggiornamento' come UpdateItem.	<a href="#">Introduzione ed esempio.</a>
DynamoDbVersionAttribute	attributo	Incrementa il numero di versione di un articolo.	<a href="#">Introduzione e discussione.</a>

<sup>1</sup> È possibile applicare un'annotazione a livello di attributo al getter o al setter, ma non a entrambi. Questa guida mostra le annotazioni sui getter.

<sup>2</sup> Il termine `property` viene normalmente utilizzato per un valore incapsulato in una classe di dati. JavaBean Tuttavia, questa guida utilizza `attribute` invece il termine, per coerenza con la terminologia utilizzata da DynamoDB.

# Lavora con Amazon EC2

Questa sezione fornisce esempi di programmazione [Amazon EC2](#) che utilizzano il AWS SDK for Java 2.x.

## Argomenti

- [Gestione delle Amazon EC2 istanze](#)
- [Zone Regioni AWS di utilizzo e disponibilità](#)
- [Collabora con i gruppi di sicurezza in Amazon EC2](#)
- [Utilizzare i metadati delle istanze Amazon EC2.](#)

## Gestione delle Amazon EC2 istanze

### Creazione di un'istanza

Crea una nuova Amazon EC2 istanza chiamando il [runInstances](#) metodo di [Ec2Client](#), fornendogli una Amazon Machine Image (AMI) [RunInstancesRequest](#) contenente l'[Amazon Machine Image \(AMI\)](#) da utilizzare e un tipo di [istanza](#).

### Importazioni

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.InstanceType;
import software.amazon.awssdk.services.ec2.model.RunInstancesRequest;
import software.amazon.awssdk.services.ec2.model.RunInstancesResponse;
import software.amazon.awssdk.services.ec2.model.Tag;
import software.amazon.awssdk.services.ec2.model.CreateTagsRequest;
import software.amazon.awssdk.services.ec2.model.Ec2Exception;
```

### Codice

```
public static String createEC2Instance(Ec2Client ec2, String name, String amiId ) {

    RunInstancesRequest runRequest = RunInstancesRequest.builder()
        .imageId(amiId)
        .instanceType(InstanceType.T1_MICRO)
        .maxCount(1)
```

```
        .minCount(1)
        .build();

RunInstancesResponse response = ec2.runInstances(runRequest);
String instanceId = response.instances().get(0).instanceId();

Tag tag = Tag.builder()
    .key("Name")
    .value(name)
    .build();

CreateTagsRequest tagRequest = CreateTagsRequest.builder()
    .resources(instanceId)
    .tags(tag)
    .build();

try {
    ec2.createTags(tagRequest);
    System.out.printf(
        "Successfully started EC2 Instance %s based on AMI %s",
        instanceId, amiId);

    return instanceId;
} catch (Ec2Exception e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}

return "";
}
```

Guarda l'[esempio completo](#) su GitHub

## Avviare un'istanza

Per avviare un'Amazon EC2 istanza, chiama il [startInstances](#) metodo di `Ec2Client`, fornendogli un file [StartInstancesRequest](#) contenente l'ID dell'istanza da avviare.

### Importazioni

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2Client;
```

```
import software.amazon.awssdk.services.ec2.model.StartInstancesRequest;
import software.amazon.awssdk.services.ec2.model.StopInstancesRequest;
```

## Codice

```
public static void startInstance(Ec2Client ec2, String instanceId) {

    StartInstancesRequest request = StartInstancesRequest.builder()
        .instanceIds(instanceId)
        .build();

    ec2.startInstances(request);
    System.out.printf("Successfully started instance %s", instanceId);
}
```

Vedi l'[esempio completo](#) su GitHub

## Arrestare un'istanza

Per fermare un'Amazon EC2istanza, chiama il [stopInstances](#) metodo di Ec2Client, fornendogli un file [StopInstancesRequest](#) contenente l'ID dell'istanza da interrompere.

## Importazioni

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.StartInstancesRequest;
import software.amazon.awssdk.services.ec2.model.StopInstancesRequest;
```

## Codice

```
public static void stopInstance(Ec2Client ec2, String instanceId) {

    StopInstancesRequest request = StopInstancesRequest.builder()
        .instanceIds(instanceId)
        .build();

    ec2.stopInstances(request);
    System.out.printf("Successfully stopped instance %s", instanceId);
}
```

Vedi l'[esempio completo](#) su GitHub

## Riavviare un'istanza

Per riavviare un'Amazon EC2istanza, chiama il [rebootInstances](#)metodo di Ec2Client, fornendogli un file [RebootInstancesRequest](#)contenente l'ID dell'istanza da riavviare.

### Importazioni

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.Ec2Exception;
import software.amazon.awssdk.services.ec2.model.RebootInstancesRequest;
```

### Codice

```
public static void rebootEC2Instance(Ec2Client ec2, String instanceId) {
    try {
        RebootInstancesRequest request = RebootInstancesRequest.builder()
            .instanceIds(instanceId)
            .build();

        ec2.rebootInstances(request);
        System.out.printf(
            "Successfully rebooted instance %s", instanceId);
    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

Vedi l'[esempio completo su](#) GitHub

## Descrivere le istanze

Per elencare le tue istanze, crea un metodo [DescribeInstancesRequest](#) chiama il metodo di Ec2Client. [describeInstances](#) Restituirà un [DescribeInstancesResponse](#)oggetto che puoi utilizzare per elencare le Amazon EC2 istanze del tuo account e della tua regione.

Le istanze sono raggruppate in base alla prenotazione. Ogni prenotazione corrisponde alla chiamata a `startInstances` che ha avviato l'istanza. Per elencare le istanze, devi prima chiamare il metodo

reservations della classe `DescribeInstancesResponse` e quindi chiamare `instances` su ciascun oggetto [Reservation](#) restituito.

## Importazioni

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.DescribeInstancesRequest;
import software.amazon.awssdk.services.ec2.model.DescribeInstancesResponse;
import software.amazon.awssdk.services.ec2.model.Instance;
import software.amazon.awssdk.services.ec2.model.Reservation;
import software.amazon.awssdk.services.ec2.model.Ec2Exception;
```

## Codice

```
public static void describeEC2Instances( Ec2Client ec2){

    String nextToken = null;

    try {

        do {

            DescribeInstancesRequest request =
DescribeInstancesRequest.builder().maxResults(6).nextToken(nextToken).build();
            DescribeInstancesResponse response = ec2.describeInstances(request);

            for (Reservation reservation : response.reservations()) {
                for (Instance instance : reservation.instances()) {
                    System.out.println("Instance Id is " + instance.instanceId());
                    System.out.println("Image id is "+ instance.imageId());
                    System.out.println("Instance type is "+
instance.instanceType());
                    System.out.println("Instance state name is "+
instance.state().name());
                    System.out.println("monitoring information is "+
instance.monitoring().state());

                }
            }

            nextToken = response.nextToken();
        } while (nextToken != null);

    } catch (Ec2Exception e) {
```

```
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

I risultati vengono paginati; puoi ottenere altri risultati passando il valore restituito dal metodo `nextToken` dell'oggetto risultante a un nuovo metodo `nextToken` dell'oggetto richiesta e utilizzando il nuovo oggetto richiesta nella chiamata successiva a `describeInstances`.

Vedi l'[esempio completo](#) su GitHub

## Monitorare un'istanza

Puoi monitorare diversi aspetti delle istanze Amazon EC2, ad esempio utilizzo della CPU e della rete, memoria disponibile e spazio su disco rimanente. Per ulteriori informazioni sul monitoraggio delle istanze, consulta [Monitoraggio Amazon EC2](#) nella Guida per l'Amazon EC2 utente delle istanze Linux.

Per iniziare a monitorare un'istanza, è necessario crearne un'[MonitorInstancesRequest](#) con l'ID dell'istanza da monitorare e passarla al metodo di `Ec2Client` [monitorInstances](#)

### Importazioni

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.MonitorInstancesRequest;
import software.amazon.awssdk.services.ec2.model.UnmonitorInstancesRequest;
```

### Codice

```
public static void monitorInstance( Ec2Client ec2, String instanceId) {

    MonitorInstancesRequest request = MonitorInstancesRequest.builder()
        .instanceIds(instanceId).build();

    ec2.monitorInstances(request);
    System.out.printf(
        "Successfully enabled monitoring for instance %s",
        instanceId);
}
```

Vedi l'[esempio completo](#) su GitHub



## Arrestare il monitoraggio delle istanze

Per interrompere il monitoraggio di un'istanza, creane un'istanza [UnmonitorInstancesRequest](#) con l'ID dell'istanza per interrompere il monitoraggio e passala al metodo di [unmonitorInstances](#) `Ec2Client`.

### Importazioni

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.MonitorInstancesRequest;
import software.amazon.awssdk.services.ec2.model.UnmonitorInstancesRequest;
```

### Codice

```
public static void unmonitorInstance(Ec2Client ec2, String instanceId) {
    UnmonitorInstancesRequest request = UnmonitorInstancesRequest.builder()
        .instanceIds(instanceId).build();

    ec2.unmonitorInstances(request);

    System.out.printf(
        "Successfully disabled monitoring for instance %s",
        instanceId);
}
```

Vedi l'[esempio completo](#) su GitHub

### Ulteriori informazioni

- [RunInstances](#) nel riferimento Amazon EC2 API
- [DescribeInstances](#) nell'Amazon EC2API Reference
- [StartInstances](#) nell'Amazon EC2API Reference
- [StopInstances](#) nell'Amazon EC2API Reference
- [RebootInstances](#) nell'Amazon EC2API Reference
- [MonitorInstances](#) nell'Amazon EC2API Reference
- [UnmonitorInstances](#) nell'Amazon EC2API Reference

## Zone Regioni AWS di utilizzo e disponibilità

### Descrivere le regioni

Per elencare le regioni disponibili per il tuo account, chiama il metodo di `Ec2Client`.

`describeRegions` Restituisce [DescribeRegionsResponse](#). Chiamare il metodo `regions` dell'oggetto restituito per ottenere un elenco di oggetti [Region](#) che rappresentano ciascuna regione.

### Importazioni

```
import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.DescribeRegionsResponse;
import software.amazon.awssdk.services.ec2.model.DescribeAvailabilityZonesResponse;
import software.amazon.awssdk.services.ec2.model.Ec2Exception;
```

### Codice

```
try {
    DescribeRegionsResponse regionsResponse = ec2.describeRegions();
    regionsResponse.regions().forEach(region -> {
        System.out.printf(
            "Found Region %s with endpoint %s%n",
            region.regionName(),
            region.endpoint());
        System.out.println();
    });
}
```

Vedi l'[esempio completo](#) su GitHub

### Descrivere le zone di disponibilità

Per elencare ogni zona di disponibilità disponibile per il tuo account, chiama il metodo di `Ec2Client`.

`describeAvailabilityZones` Restituisce [DescribeAvailabilityZonesResponse](#). Chiama il suo `availabilityZones` metodo per ottenere un elenco di [AvailabilityZone](#) oggetti che rappresentano ogni zona di disponibilità.

### Importazioni

```
import software.amazon.awssdk.services.ec2.Ec2Client;
```

```
import software.amazon.awssdk.services.ec2.model.DescribeRegionsResponse;
import software.amazon.awssdk.services.ec2.model.DescribeAvailabilityZonesResponse;
import software.amazon.awssdk.services.ec2.model.Ec2Exception;
```

## Codice

### Crea Ec2Client.

```
software.amazon.awssdk.regions.Region region =
software.amazon.awssdk.regions.Region.US_EAST_1;
Ec2Client ec2 = Ec2Client.builder()
    .region(region)
    .build();
```

Quindi chiama `describeAvailabilityZones ()` e recupera i risultati.

```
DescribeAvailabilityZonesResponse zonesResponse =
ec2.describeAvailabilityZones();
zonesResponse.availabilityZones().forEach(zone -> {
    System.out.printf(
        "Found Availability Zone %s with status %s in region %s\n",
        zone.zoneName(),
        zone.state(),
        zone.regionName()
    );
    System.out.println();
});
```

Vedi l'[esempio completo](#) su GitHub

## Descrivere gli account

Per elencare le informazioni relative a EC2 sul tuo account, chiama il metodo di `Ec2Client`. `describeAccountAttributes` Questo metodo restituisce un oggetto.

[DescribeAccountAttributesResponse](#) Invoca questo `accountAttributes` metodo degli oggetti per ottenere un elenco di [AccountAttribute](#) oggetti. È possibile scorrere l'elenco per recuperare un oggetto. `AccountAttribute`

Puoi ottenere i valori degli attributi del tuo account richiamando il metodo dell'`AccountAttribute` oggetto. `attributeValues` Questo metodo restituisce un elenco di

[AccountAttributeValue](#) oggetti. È possibile scorrere questo secondo elenco per visualizzare il valore degli attributi (vedere l'esempio di codice riportato di seguito).

## Importazioni

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.DescribeAccountAttributesResponse;
import software.amazon.awssdk.services.ec2.model.Ec2Exception;
```

## Codice

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.DescribeAccountAttributesResponse;
import software.amazon.awssdk.services.ec2.model.Ec2Exception;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DescribeAccount {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        Ec2Client ec2 = Ec2Client.builder()
            .region(region)
            .build();

        describeEC2Account(ec2);
        System.out.print("Done");
        ec2.close();
    }

    public static void describeEC2Account(Ec2Client ec2) {
        try {
            DescribeAccountAttributesResponse accountResults =
ec2.describeAccountAttributes();
            accountResults.accountAttributes().forEach(attribute -> {
```

```
        System.out.print("\n The name of the attribute is " +
attribute.attributeName());
        attribute.attributeValues().forEach(
            myValue -> System.out.print("\n The value of the attribute is "
+ myValue.attributeValue()));
    });

    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

Vedi l'[esempio completo](#) su GitHub.

## Ulteriori informazioni

- [Regioni e zone di disponibilità](#) nella Guida per l' Amazon EC2 utente per le istanze Linux
- [DescribeRegions](#) nel riferimento Amazon EC2 API
- [DescribeAvailabilityZones](#) nell' Amazon EC2 API Reference

## Collabora con i gruppi di sicurezza in Amazon EC2

### Creazione di un gruppo di sicurezza

Per creare un gruppo di sicurezza, chiama il `createSecurityGroup` metodo di `Ec2Client` con un [CreateSecurityGroupRequest](#) che contiene il nome della chiave.

### Importazioni

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.CreateSecurityGroupRequest;
import software.amazon.awssdk.services.ec2.model.AuthorizeSecurityGroupIngressRequest;
import software.amazon.awssdk.services.ec2.model.AuthorizeSecurityGroupIngressResponse;
import software.amazon.awssdk.services.ec2.model.Ec2Exception;
import software.amazon.awssdk.services.ec2.model.IpPermission;
import software.amazon.awssdk.services.ec2.model.CreateSecurityGroupResponse;
import software.amazon.awssdk.services.ec2.model.IpRange;
```

## Codice

```
        CreateSecurityGroupRequest createRequest =
CreateSecurityGroupRequest.builder()
    .groupName(groupName)
    .description(groupDesc)
    .vpcId(vpcId)
    .build();

        CreateSecurityGroupResponse resp= ec2.createSecurityGroup(createRequest);
```

Vedi l'[esempio completo](#) su. GitHub

## Configurare un gruppo di sicurezza

Un gruppo di sicurezza può controllare sia il traffico in entrata (ingresso) che quello in uscita (in uscita) verso le tue istanze. Amazon EC2

Per aggiungere regole di ingresso al tuo gruppo di sicurezza, usa il `authorizeSecurityGroupIngress` metodo di `Ec2Client`, fornendo il nome del gruppo di sicurezza e le regole di accesso ([IpPermission](#)) che desideri assegnargli all'interno di un oggetto. [AuthorizeSecurityGroupIngressRequest](#) Nell'esempio seguente viene mostrato come aggiungere autorizzazioni IP a un gruppo di sicurezza.

## Importazioni

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.CreateSecurityGroupRequest;
import software.amazon.awssdk.services.ec2.model.AuthorizeSecurityGroupIngressRequest;
import software.amazon.awssdk.services.ec2.model.AuthorizeSecurityGroupIngressResponse;
import software.amazon.awssdk.services.ec2.model.Ec2Exception;
import software.amazon.awssdk.services.ec2.model.IpPermission;
import software.amazon.awssdk.services.ec2.model.CreateSecurityGroupResponse;
import software.amazon.awssdk.services.ec2.model.IpRange;
```

## Codice

Innanzitutto, crea un `Ec2Client`

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

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

Quindi usa il metodo di `Ec2Client`, `authorizeSecurityGroupIngress`

```
IpRange ipRange = IpRange.builder()
    .cidrIp("0.0.0.0/0").build();

IpPermission ipPerm = IpPermission.builder()
    .ipProtocol("tcp")
    .toPort(80)
    .fromPort(80)
    .ipRanges(ipRange)
    .build();

IpPermission ipPerm2 = IpPermission.builder()
    .ipProtocol("tcp")
    .toPort(22)
    .fromPort(22)
    .ipRanges(ipRange)
    .build();

AuthorizeSecurityGroupIngressRequest authRequest =
    AuthorizeSecurityGroupIngressRequest.builder()
        .groupName(groupName)
        .ipPermissions(ipPerm, ipPerm2)
        .build();

AuthorizeSecurityGroupIngressResponse authResponse =
    ec2.authorizeSecurityGroupIngress(authRequest);

System.out.printf(
    "Successfully added ingress policy to Security Group %s",
    groupName);

return resp.groupId();

} catch (Ec2Exception e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
return "";
}
```

Per aggiungere una regola di uscita al gruppo di sicurezza, fornisci dati simili in un metodo di [AuthorizeSecurityGroupEgressRequest](#) `Ec2Client.authorizeSecurityGroupEgress`

Vedi l'esempio [completo](#) su GitHub

## Descrivere i gruppi di sicurezza

Per descrivere i tuoi gruppi di sicurezza o ottenere informazioni su di essi, chiama il metodo di `describeSecurityGroups` `Ec2Client`. Restituisce un [DescribeSecurityGroupsResponse](#) comando che puoi usare per accedere all'elenco dei gruppi di sicurezza chiamando il relativo `securityGroups` metodo, che restituisce un elenco di [SecurityGroup](#) oggetti.

### Importazioni

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.DescribeSecurityGroupsRequest;
import software.amazon.awssdk.services.ec2.model.DescribeSecurityGroupsResponse;
import software.amazon.awssdk.services.ec2.model.SecurityGroup;
import software.amazon.awssdk.services.ec2.model.Ec2Exception;
```

### Codice

```
public static void describeEC2SecurityGroups(Ec2Client ec2, String groupId) {

    try {
        DescribeSecurityGroupsRequest request =
            DescribeSecurityGroupsRequest.builder()
                .groupIds(groupId).build();

        DescribeSecurityGroupsResponse response =
            ec2.describeSecurityGroups(request);

        for(SecurityGroup group : response.securityGroups()) {
            System.out.printf(
                "Found Security Group with id %s, " +
                "vpc id %s " +
                "and description %s",
                group.groupId(),
                group.vpcId(),
                group.description());
        }
    }
}
```



```
    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

Vedi l'[esempio completo](#) su GitHub.

## Eliminare un gruppo di sicurezza

Per eliminare un gruppo di sicurezza, chiama il `deleteSecurityGroup` metodo di `Ec2Client`, passandogli un gruppo di sicurezza [DeleteSecurityGroupRequest](#) che contiene l'ID del gruppo di sicurezza da eliminare.

### Importazioni

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.DeleteSecurityGroupRequest;
import software.amazon.awssdk.services.ec2.model.Ec2Exception;
```

### Codice

```
public static void deleteEC2SecGroup(Ec2Client ec2, String groupId) {

    try {
        DeleteSecurityGroupRequest request = DeleteSecurityGroupRequest.builder()
            .groupId(groupId)
            .build();

        ec2.deleteSecurityGroup(request);
        System.out.printf(
            "Successfully deleted Security Group with id %s", groupId);
    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

Vedi l'[esempio completo](#) su GitHub

## Ulteriori informazioni

- [Amazon EC2 Gruppi di sicurezza](#) nella Guida per l' Amazon EC2 utente per le istanze Linux
- [Autorizza il traffico in entrata per le tue istanze Linux nella Guida per l' Amazon EC2 utente per le istanze Linux](#)
- [CreateSecurityGroup](#) nella guida di riferimento alle API Amazon EC2
- [DescribeSecurityGroups](#) nell' Amazon EC2 API Reference
- [DeleteSecurityGroup](#) nell' Amazon EC2 API Reference
- [AuthorizeSecurityGroupIngress](#) nell' Amazon EC2 API Reference

## Utilizzare i metadati delle istanze Amazon EC2.

Un client Java SDK per Amazon EC2 Instance Metadata Service (client di metadati) consente alle applicazioni di accedere ai metadati sulla loro istanza EC2 locale. Il client di metadati funziona con l'istanza locale di [IMDSv2](#) (Instance Metadata Service v2) e utilizza richieste orientate alla sessione.

Nell'SDK sono disponibili due classi client. La modalità sincrona [Ec2MetadataClient](#) serve per bloccare le operazioni e [Ec2MetadataAsyncClient](#) quella per i casi d'uso asincroni e non bloccanti.

## Nozioni di base

Per utilizzare il client di metadati, aggiungi l'artefatto `imds` Maven al tuo progetto. Sono necessarie anche le classi per un [SdkHttpClient](#) (o un [SdkAsyncHttpClient](#) per la variante asincrona) sul classpath.

Il seguente Maven XML mostra i frammenti di dipendenza per l'utilizzo del client sincrono [URLConnectionHttpClient](#) insieme alla dipendenza per i client di metadati.

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

```

    </dependencies>
</dependencyManagement>

<dependencies>
  <dependency>
    <groupId>software.amazon.awssdk</groupId>
    <artifactId>imds</artifactId>
  </dependency>
  <dependency>
    <groupId>software.amazon.awssdk</groupId>
    <artifactId>url-connection-client</artifactId>
  </dependency>
  <!-- other dependencies -->
</dependencies>

```

Cerca nel [repository centrale di Maven](#) la versione più recente dell'`bomartefatto`.

Per utilizzare un client HTTP asincrono, sostituisci il frammento di dipendenza per l'`url-connection-client`artefatto. Ad esempio, il seguente frammento di codice introduce l'[NettyNioAsyncHttpClient](#)implementazione.

```

<dependency>
  <groupId>software.amazon.awssdk</groupId>
  <artifactId>netty-nio-client</artifactId>
</dependency>

```

## Usa il client di metadati

Crea un'istanza di un client di metadati

È possibile creare un'istanza di un sistema sincrono `Ec2MetadataClient` quando nel classpath è presente una sola implementazione dell'`SdkHttpClient`interfaccia. Per fare ciò, chiama il `Ec2MetadataClient#create()` metodo statico come visualizzato nel frammento seguente.

```

Ec2MetadataClient client = Ec2MetadataClient.create(); //
'Ec2MetadataAsyncClient#create' is the asynchronous version.

```

Se l'applicazione ha più implementazioni dell'`SdkHttpClient`interfaccia `SdkHttpClient` or, è necessario specificare un'implementazione da utilizzare per il client di metadati, come mostrato nella [the section called “Client HTTP configurabile”](#) sezione.

**Note**

Per la maggior parte dei client di servizi, come Amazon S3, l'SDK for Java aggiunge automaticamente le implementazioni dell'`SdkHttpAsyncClient` interfaccia `SdkHttpClient` or. Se il tuo client di metadati utilizza la stessa implementazione, `Ec2MetadataClient#create()` funzionerà. Se è necessaria un'implementazione diversa, è necessario specificarla quando si crea il client di metadati.

## Invia richieste

Per recuperare i metadati dell'istanza, create un'istanza della `EC2MetadataClient` classe e chiamate il `get` metodo con un parametro `path` che specifica la [categoria dei metadati dell'istanza](#).

L'esempio seguente stampa il valore associato alla `ami-id` chiave sulla console.

```
Ec2MetadataClient client = Ec2MetadataClient.create();
Ec2MetadataResponse response = client.get("/latest/meta-data/ami-id");
System.out.println(response.asString());
client.close(); // Closes the internal resources used by the Ec2MetadataClient class.
```

Se il percorso non è valido, il `get` metodo genera un'eccezione.

Riutilizza la stessa istanza client per più richieste, ma richiama `close` il client quando non è più necessario rilasciare risorse. Dopo aver chiamato il metodo `close`, l'istanza client non può più essere utilizzata.

## Analizza le risposte

I metadati delle istanze EC2 possono essere emessi in diversi formati. Testo normale e JSON sono i formati più comunemente usati. I client di metadati offrono modi per lavorare con questi formati.

Come mostra l'esempio seguente, usa il `asString` metodo per ottenere i dati come stringa Java. È inoltre possibile utilizzare il `asList` metodo per separare una risposta in testo normale che restituisce più righe.

```
Ec2MetadataClient client = Ec2MetadataClient.create();
Ec2MetadataResponse response = client.get("/latest/meta-data/");
String fullResponse = response.asString();
```

```
List<String> splits = response.asList();
```

Se la risposta è in JSON, utilizza il `Ec2MetadataResponse#asDocument` metodo per analizzare la risposta JSON in un'istanza [Document](#), come mostrato nel seguente frammento di codice.

```
Document fullResponse = response.asDocument();
```

Verrà generata un'eccezione se il formato dei metadati non è in JSON. Se la risposta viene analizzata correttamente, puoi utilizzare l'[API del documento](#) per ispezionare la risposta in modo più dettagliato. Consulta il [grafico delle categorie dei metadati](#) delle istanze per scoprire quali categorie di metadati forniscono risposte in formato JSON.

## Configurazione di un client di metadati

### Tentativi

È possibile configurare un client di metadati con un meccanismo di riprova. In tal caso, il client può riprovare automaticamente le richieste che falliscono per motivi imprevisti. Per impostazione predefinita, il client riprova tre volte su una richiesta non riuscita con un tempo di backoff esponenziale tra i tentativi.

Se il tuo caso d'uso richiede un meccanismo di ripetizione diverso, puoi personalizzare il client utilizzando il `retryPolicy` metodo del relativo generatore. Ad esempio, l'esempio seguente mostra un client sincrono configurato con un ritardo fisso di due secondi tra i tentativi e cinque tentativi di ripetizione.

```
BackoffStrategy fixedBackoffStrategy =
    FixedDelayBackoffStrategy.create(Duration.ofSeconds(2));
Ec2MetadataClient client =
    Ec2MetadataClient.builder()
        .retryPolicy(retryPolicyBuilder ->
            retryPolicyBuilder.numRetries(5)

            .backoffStrategy(fixedBackoffStrategy))
        .build();
```

Ce ne sono diversi [BackoffStrategies](#) che puoi usare con un client di metadati.

Puoi anche disabilitare completamente il meccanismo di ripetizione, come mostra il frammento seguente.

```
Ec2MetadataClient client =
    Ec2MetadataClient.builder()
        .retryPolicy(Ec2MetadataRetryPolicy.none())
        .build();
```

L'utilizzo `Ec2MetadataRetryPolicy#none()` disabilita la politica di ripetizione predefinita in modo che il client di metadati non tenti nuovi tentativi.

## Versione IP

Per impostazione predefinita, un client di metadati utilizza l'endpoint IPv4 all'indirizzo `http://169.254.169.254`. Per modificare il client in modo che utilizzi la versione IPv6, utilizza il metodo `endpointMode` o il `endpoint` metodo del generatore. Si verifica un'eccezione se entrambi i metodi vengono chiamati sul builder.

Negli esempi seguenti vengono mostrate entrambe le opzioni IPv6.

```
Ec2MetadataClient client =
    Ec2MetadataClient.builder()
        .endpointMode(EndpointMode.IPV6)
        .build();
```

```
Ec2MetadataClient client =
    Ec2MetadataClient.builder()
        .endpoint(URI.create("http://[fd00:ec2::254]"))
        .build();
```

## Caratteristiche principali

### Client asincrona

Per utilizzare la versione non bloccante del client, create un'istanza della `Ec2MetadataAsyncClient` classe. Il codice nell'esempio seguente crea un client asincrono con impostazioni predefinite e utilizza il `get` metodo per recuperare il valore della `ami-id` chiave.

```
Ec2MetadataAsyncClient asyncClient = Ec2MetadataAsyncClient.create();
CompletableFuture<Ec2MetadataResponse> response = asyncClient.get("/latest/meta-data/ami-id");
```

Il valore `java.util.concurrent.CompletableFuture` restituito dal `get` metodo viene completato quando viene restituita la risposta. L'esempio seguente stampa `iami-id` metadati sulla console.

```
response.thenAccept(metadata -> System.out.println(metadata.asString()));
```

## Client HTTP configurabile

Il generatore di ogni client di metadati dispone di un `httpClient` metodo che è possibile utilizzare per fornire un client HTTP personalizzato.

L'esempio seguente mostra il codice per un' `URLConnectionHttpClient` istanza personalizzata.

```
SdkHttpClient httpClient =
    UrlConnectionHttpClient.builder()
        .socketTimeout(Duration.ofMinutes(5))
        .proxyConfiguration(proxy ->
            proxy.endpoint(URI.create("http://proxy.example.net:8888")))
        .build();
Ec2MetadataClient metaDataClient =
    Ec2MetadataClient.builder()
        .httpClient(httpClient)
        .build();
// Use the metaDataClient instance.
metaDataClient.close(); // Close the instance when no longer needed.
```

L'esempio seguente mostra il codice per un' `NettyNioAsyncHttpClient` istanza personalizzata con un client di metadati asincrono.

```
SdkAsyncHttpClient httpAsyncClient =
    NettyNioAsyncHttpClient.builder()
        .connectionTimeout(Duration.ofMinutes(5))
        .maxConcurrency(100)
        .build();
Ec2MetadataAsyncClient asyncMetaDataClient =
    Ec2MetadataAsyncClient.builder()
        .httpClient(httpAsyncClient)
        .build();
// Use the asyncMetaDataClient instance.
asyncMetaDataClient.close(); // Close the instance when no longer needed.
```

L'[the section called “Client HTTP”](#) argomento di questa guida fornisce dettagli su come configurare i client HTTP disponibili nell'SDK for Java.

## Memorizzazione nella cache dei token

Poiché i client di metadati utilizzano IMDSv2, tutte le richieste sono associate a una sessione. Una sessione è definita da un token con una scadenza, che il client di metadati gestisce per te. Ogni richiesta di metadati riutilizza automaticamente il token fino alla sua scadenza.

Per impostazione di default, un token dura sei ore (21.600 secondi). Ti consigliamo di mantenere il time-to-live valore predefinito, a meno che il caso d'uso specifico non richieda una configurazione avanzata.

Se necessario, configura la durata utilizzando il `tokenTtl` builder. Ad esempio, il codice nel seguente frammento di codice crea un client con una durata della sessione di cinque minuti.

```
Ec2MetadataClient client =  
    Ec2MetadataClient.builder()  
        .tokenTtl(Duration.ofMinutes(5))  
        .build();
```

Se si omette di chiamare il `tokenTtl` metodo sul builder, viene invece utilizzata la durata predefinita di 21.600.

## Utilizzo delle IAM

Questa sezione fornisce esempi di programmazione AWS Identity and Access Management (IAM) utilizzando la versione AWS SDK for Java 2.x.

AWS Identity and Access Management (IAM) consente di controllare in modo sicuro l'accesso ai AWS servizi e alle risorse per gli utenti. Utilizzando IAM, è possibile creare e gestire AWS utenti e gruppi e utilizzare le autorizzazioni per consentire e negare il loro accesso alle risorse. AWS Per una guida completa IAM, consulta la Guida per l'[IAM utente](#).

Gli esempi seguenti includono solo il codice necessario per dimostrare ciascuna tecnica. Il [codice di esempio completo è disponibile su GitHub](#). Da qui puoi scaricare un singolo file sorgente o clonare l'archivio localmente per ottenere tutti gli esempi da creare ed eseguire.

### Argomenti

- [Gestione delle chiavi di IAM accesso](#)



- [Gestisci IAM gli utenti](#)
- [Crea policy IAM con AWS SDK for Java 2.x](#)
- [Lavora con IAM le politiche](#)
- [Lavora con i certificati IAM del server](#)

## Gestione delle chiavi di IAM accesso

### Creare una chiave di accesso

Per creare una chiave di IAM accesso, chiama il `IamClient`'s `createAccessKey` metodo con un [CreateAccessKeyRequest](#) oggetto.

#### Note

È necessario impostare la regione su `AWS_GLOBAL` affinché `IamClient` le chiamate funzionino perché IAM è un servizio globale.

### Importazioni

```
import software.amazon.awssdk.services.iam.model.CreateAccessKeyRequest;
import software.amazon.awssdk.services.iam.model.CreateAccessKeyResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;
```

### Codice

```
public static String createIAMAccessKey(IamClient iam, String user) {

    try {
        CreateAccessKeyRequest request = CreateAccessKeyRequest.builder()
            .userName(user).build();

        CreateAccessKeyResponse response = iam.createAccessKey(request);
        String keyId = response.accessKey().accessKeyId();
        return keyId;

    } catch (IamException e) {
```

```
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
```

Guarda l'esempio [completo](#) su. GitHub

## Elencare le chiavi di accesso

Per elencare le chiavi di accesso per un determinato utente, create un [ListAccessKeysRequest](#) oggetto che contenga il nome utente per cui elencare le chiavi e passatelo al `IamClient`'s `listAccessKeys` metodo.

### Note

Se non fornite un nome utente a `listAccessKeys`, tenterà di elencare le chiavi di accesso associate a chi Account AWS ha firmato la richiesta.

## Importazioni

```
import software.amazon.awssdk.services.iam.model.AccessKeyMetadata;
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.services.iam.model.ListAccessKeysRequest;
import software.amazon.awssdk.services.iam.model.ListAccessKeysResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
```

## Codice

```
public static void listKeys( IamClient iam,String userName ){

    try {
        boolean done = false;
        String newMarker = null;

        while (!done) {
            ListAccessKeysResponse response;

            if(newMarker == null) {
```

```

        ListAccessKeysRequest request = ListAccessKeysRequest.builder()
            .userName(userName).build();
        response = iam.listAccessKeys(request);
    } else {
        ListAccessKeysRequest request = ListAccessKeysRequest.builder()
            .userName(userName)
            .marker(newMarker).build();
        response = iam.listAccessKeys(request);
    }

    for (AccessKeyMetadata metadata :
        response.accessKeyMetadata()) {
        System.out.format("Retrieved access key %s",
            metadata.accessKeyId());
    }

    if (!response.isTruncated()) {
        done = true;
    } else {
        newMarker = response.marker();
    }
}

} catch (IamException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}

```

I risultati di `listAccessKeys` sono paginati (con un massimo predefinito di 100 record per chiamata). È possibile richiamare `isTruncated` l'[ListAccessKeysResponse](#) oggetto restituito per verificare se la query ha restituito un numero inferiore di risultati rispetto a quelli disponibili. In tal caso, chiama `marker` su `ListAccessKeysResponse` e utilizzalo durante la creazione di una nuova richiesta. Utilizza la nuova richiesta nella prossima invocazione di `listAccessKeys`.

Vedi l'[esempio completo](#) su GitHub.

## Recupero dell'ora ultimo utilizzo di una chiave di accesso

Per sapere l'ora in cui è stata utilizzata l'ultima volta una chiave di accesso, chiamate il `IamClient`'s `getAccessKeyLastUsed` metodo con l'ID della chiave di accesso (che può essere passato utilizzando un [GetAccessKeyLastUsedRequest](#) oggetto).

È quindi possibile utilizzare l'[GetAccessKeyLastUsedResponse](#) oggetto restituito per recuperare l'ultima ora utilizzata della chiave.

## Importazioni

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.GetAccessKeyLastUsedRequest;
import software.amazon.awssdk.services.iam.model.GetAccessKeyLastUsedResponse;
import software.amazon.awssdk.services.iam.model.IamException;
```

## Codice

```
public static void getAccessKeyLastUsed(IamClient iam, String accessId ){

    try {
        GetAccessKeyLastUsedRequest request = GetAccessKeyLastUsedRequest.builder()
            .accessKeyId(accessId).build();

        GetAccessKeyLastUsedResponse response = iam.getAccessKeyLastUsed(request);

        System.out.println("Access key was last used at: " +
            response.accessKeyLastUsed().lastUsedDate());

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    System.out.println("Done");
}
```

Vedi l'[esempio completo](#) su. GitHub

## Attivare o disattivare le chiavi di accesso

È possibile attivare o disattivare una chiave di accesso creando un [UpdateAccessKeyRequest](#) oggetto, fornendo l'ID della chiave di accesso, facoltativamente il nome utente e quello desiderato [status](#), quindi passando l'oggetto della richiesta al `IamClient`'s `updateAccessKey` metodo.

## Importazioni

```
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.services.iam.model.StatusType;
import software.amazon.awssdk.services.iam.model.UpdateAccessKeyRequest;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
```

## Codice

```
public static void updateKey(IamClient iam, String username, String accessId,
String status ) {

    try {
        if (status.toLowerCase().equalsIgnoreCase("active")) {
            statusType = StatusType.ACTIVE;
        } else if (status.toLowerCase().equalsIgnoreCase("inactive")) {
            statusType = StatusType.INACTIVE;
        } else {
            statusType = StatusType.UNKNOWN_TO_SDK_VERSION;
        }
        UpdateAccessKeyRequest request = UpdateAccessKeyRequest.builder()
            .accessKeyId(accessId)
            .userName(username)
            .status(statusType)
            .build();

        iam.updateAccessKey(request);

        System.out.printf(
            "Successfully updated the status of access key %s to" +
            "status %s for user %s", accessId, status, username);

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

Vedi l'[esempio completo](#) su GitHub

## Eliminare una chiave di accesso

Per eliminare definitivamente una chiave di accesso, chiamate il `IamClient`'s `deleteKey` metodo, fornendogli un codice [DeleteAccessKeyRequest](#) contenente l'ID e il nome utente della chiave di accesso.

### Note

Dopo che è stata eliminata, una chiave non può più essere recuperata né utilizzata. Per disattivare temporaneamente una chiave in modo che possa essere riattivata in un secondo momento, usa invece [updateAccessKey](#) method.

## Importazioni

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.DeleteAccessKeyRequest;
import software.amazon.awssdk.services.iam.model.IamException;
```

## Codice

```
public static void deleteKey(IamClient iam ,String username, String accessKey ) {

    try {
        DeleteAccessKeyRequest request = DeleteAccessKeyRequest.builder()
            .accessKeyId(accessKey)
            .userName(username)
            .build();

        iam.deleteAccessKey(request);
        System.out.println("Successfully deleted access key " + accessKey +
            " from user " + username);

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

Vedi l'[esempio completo](#) su. GitHub

## Ulteriori informazioni

- [CreateAccessKey](#) nel riferimento IAM API
- [ListAccessKeys](#) nell' IAM API Reference
- [GetAccessKeyLastUsed](#) nell' IAM API Reference
- [UpdateAccessKey](#) nell' IAM API Reference
- [DeleteAccessKey](#) nell' IAM API Reference

## Gestisci IAM gli utenti

### Creazione di un utente

Crea un nuovo IAM utente fornendo il nome utente al `createUser` metodo `IamClient`'s utilizzando un [CreateUserRequest](#) oggetto contenente il nome utente.

### Importazioni

```
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.services.iam.model.CreateUserRequest;
import software.amazon.awssdk.services.iam.model.CreateUserResponse;
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.waiters.IamWaiter;
import software.amazon.awssdk.services.iam.model.GetUserRequest;
import software.amazon.awssdk.services.iam.model.GetUserResponse;
```

### Codice

```
public static String createIAMUser(IamClient iam, String username ) {

    try {
        // Create an IamWaiter object
        IamWaiter iamWaiter = iam.waiter();

        CreateUserRequest request = CreateUserRequest.builder()
            .userName(username)
            .build();

        CreateUserResponse response = iam.createUser(request);
```

```
// Wait until the user is created
GetUserRequest userRequest = GetUserRequest.builder()
    .userName(response.user().userName())
    .build();

WaiterResponse<GetUserResponse> waitUntilUserExists =
iamWaiter.waitUntilUserExists(userRequest);
waitUntilUserExists.matched().response().ifPresent(System.out::println);
return response.user().userName();

} catch (IamException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
return "";
}
```

Vedi [l'esempio completo](#) su GitHub.

## Creazione di elenchi sugli utenti

Per elencare gli IAM utenti del tuo account, creane uno nuovo [ListUsersRequeste](#) passalo al `listUsers` metodo `IamClient`'s. È possibile recuperare l'elenco degli utenti `users` chiamando l'[ListUsersResponse](#) oggetto restituito.

L'elenco di utenti restituito da `listUsers` è paginato. È possibile verificare se ci sono più risultati da recuperare chiamando il metodo `isTruncated` dell'oggetto di risposta. Se viene restituito `true`, chiamare il metodo `marker()` dell'oggetto di risposta. Utilizza il valore del contrassegno per creare un nuovo oggetto di risposta. Quindi, chiama nuovamente il metodo `listUsers` con una nuova richiesta.

## Importazioni

```
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.services.iam.model.ListUsersRequest;
import software.amazon.awssdk.services.iam.model.ListUsersResponse;
import software.amazon.awssdk.services.iam.model.User;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
```

## Codice



```
public static void listAllUsers(IamClient iam ) {

    try {

        boolean done = false;
        String newMarker = null;

        while(!done) {
            ListUsersResponse response;

            if (newMarker == null) {
                ListUsersRequest request = ListUsersRequest.builder().build();
                response = iam.listUsers(request);
            } else {
                ListUsersRequest request = ListUsersRequest.builder()
                    .marker(newMarker).build();
                response = iam.listUsers(request);
            }

            for(User user : response.users()) {
                System.out.format("\n Retrieved user %s", user.userName());
            }

            if(!response.isTruncated()) {
                done = true;
            } else {
                newMarker = response.marker();
            }
        }
    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

Vedi l'[esempio completo](#) su. GitHub

## Aggiorna un utente

Per aggiornare un utente, chiamate il `updateUser` metodo dell' `IamClient` oggetto, che accetta un [UpdateUserRequest](#) oggetto che potete usare per modificare il nome o il percorso dell'utente.

## Importazioni

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.services.iam.model.UpdateUserRequest;
```

## Codice

```
public static void updateIAMUser(IamClient iam, String curName,String newName ) {

    try {
        UpdateUserRequest request = UpdateUserRequest.builder()
            .userName(curName)
            .newUserName(newName)
            .build();

        iam.updateUser(request);
        System.out.printf("Successfully updated user to username %s",
            newName);
    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

Vedi l'[esempio completo](#) su GitHub.

## Eliminazione di un utente

Per eliminare un utente, chiama la `deleteUser` richiesta `IamClient`'s con un [UpdateUserRequest](#) oggetto impostato con il nome utente da eliminare.

## Importazioni

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.DeleteUserRequest;
import software.amazon.awssdk.services.iam.model.IamException;
```

## Codice

```
public static void deleteIAMUser(IamClient iam, String userName) {
```

```
try {
    DeleteUserRequest request = DeleteUserRequest.builder()
        .userName(userName)
        .build();

    iam.deleteUser(request);
    System.out.println("Successfully deleted IAM user " + userName);
} catch (IamException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
```

Vedi l'[esempio completo](#) su GitHub.

## Ulteriori informazioni

- [IAMUtenti](#) nella Guida per l'IAMutente
- [Gestione IAM degli utenti](#) nella Guida IAM per l'utente
- [CreateUser](#) nel riferimento alle IAM API
- [ListUsers](#) nell'IAMAPI Reference
- [UpdateUser](#) nell'IAMAPI Reference
- [DeleteUser](#) nell'IAMAPI Reference

## Crea policy IAM con AWS SDK for Java 2.x

L'[API IAM Policy Builder](#) è una libreria che puoi utilizzare per creare [policy IAM](#) in Java e caricarle su AWS Identity and Access Management (IAM).

Invece di creare una policy IAM assemblando manualmente una stringa JSON o leggendo un file, l'API fornisce un approccio lato client orientato agli oggetti per generare la stringa JSON. Quando leggi una policy IAM esistente in formato JSON, l'API la converte in un'istanza per la gestione.

### [IamPolicy](#)

L'API IAM Policy Builder è diventata disponibile con la versione 2.20.105 dell'SDK, quindi usa quella versione o una successiva nel tuo file di build di Maven. [Il numero di versione più recente dell'SDK è elencato su Maven Central.](#)

Il seguente frammento mostra un esempio di blocco di dipendenza per un file Maven. `pom.xml` Ciò ti consente di utilizzare l'API IAM Policy Builder nel tuo progetto.

```
<dependency>
  <groupId>software.amazon.awssdk</groupId>
  <artifactId>iam-policy-builder</artifactId>
  <version>2.20.139</version>
</dependency>
```

## Creazione di un **IamPolicy**

Questa sezione mostra diversi esempi di come creare policy utilizzando l'API IAM Policy Builder.

In ciascuno degli esempi seguenti, inizia con [IamPolicy.Builder](#) e aggiungi una o più istruzioni utilizzando il `addStatement` metodo. Seguendo questo schema, [IamStatement.Builder](#) dispone di metodi per aggiungere effetti, azioni, risorse e condizioni all'istruzione.

Esempio: creare una politica basata sul tempo

L'esempio seguente crea una policy basata sull'identità che consente l'azione di Amazon DynamoDB tra due momenti temporali. `GetItem`

```
public String timeBasedPolicyExample() {
    IamPolicy policy = IamPolicy.builder()
        .addStatement(b -> b
            .effect(IamEffect.ALLOW)
            .addAction("dynamodb:GetItem")
            .addResource(IamResource.ALL)
            .addCondition(b1 -> b1
                .operator(IamConditionOperator.DATE_GREATER_THAN)
                .key("aws:CurrentTime")
                .value("2020-04-01T00:00:00Z"))
            .addCondition(b1 -> b1
                .operator(IamConditionOperator.DATE_LESS_THAN)
                .key("aws:CurrentTime")
                .value("2020-06-30T23:59:59Z")))
        .build();

    // Use an IamPolicyWriter to write out the JSON string to a more readable
    format.
    return policy.toJson(IamPolicyWriter.builder()
        .prettyPrint(true)
```

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

## Output JSON

L'ultima istruzione dell'esempio precedente restituisce la seguente stringa JSON.

Ulteriori informazioni su questo [esempio sono disponibili](#) nella Guida per l'AWS Identity and Access Management utente.

```
{
  "Version" : "2012-10-17",
  "Statement" : {
    "Effect" : "Allow",
    "Action" : "dynamodb:GetItem",
    "Resource" : "*",
    "Condition" : {
      "DateGreaterThan" : {
        "aws:CurrentTime" : "2020-04-01T00:00:00Z"
      },
      "DateLessThan" : {
        "aws:CurrentTime" : "2020-06-30T23:59:59Z"
      }
    }
  }
}
```

## Esempio: specificare più condizioni

L'esempio seguente mostra come creare una policy basata sull'identità che consenta l'accesso a specifici attributi di DynamoDB. La policy contiene due condizioni.

```
public String multipleConditionsExample() {
    IamPolicy policy = IamPolicy.builder()
        .addStatement(b -> b
            .effect(IamEffect.ALLOW)
            .addAction("dynamodb:GetItem")
            .addAction("dynamodb:BatchGetItem")
            .addAction("dynamodb:Query")
            .addAction("dynamodb:PutItem")
            .addAction("dynamodb:UpdateItem")
            .addAction("dynamodb>DeleteItem")
            .addAction("dynamodb:BatchWriteItem")
```

```

        .addResource("arn:aws:dynamodb:*:*:table/table-name")

        .addConditions(IamConditionOperator.STRING_EQUALS.addPrefix("ForAllValues:"),
            "dynamodb:Attributes",
            List.of("column-name1", "column-name2", "column-
name3"))

            .addCondition(b1 ->
b1.operator(IamConditionOperator.STRING_EQUALS.addSuffix("IfExists"))
            .key("dynamodb:Select")
            .value("SPECIFIC_ATTRIBUTES"))

        .build();

    return policy.toJson(IamPolicyWriter.builder()
        .prettyPrint(true).build());
}

```

## Output JSON

L'ultima istruzione dell'esempio precedente restituisce la seguente stringa JSON.

Ulteriori informazioni su questo [esempio sono disponibili](#) nella Guida per l'AWS Identity and Access Management utente.

```

{
  "Version" : "2012-10-17",
  "Statement" : {
    "Effect" : "Allow",
    "Action" : [ "dynamodb:GetItem", "dynamodb:BatchGetItem", "dynamodb:Query",
"dynamodb:PutItem", "dynamodb:UpdateItem", "dynamodb>DeleteItem",
"dynamodb:BatchWriteItem" ],
    "Resource" : "arn:aws:dynamodb:*:*:table/table-name",
    "Condition" : {
      "ForAllValues:StringEquals" : {
        "dynamodb:Attributes" : [ "column-name1", "column-name2", "column-name3" ]
      },
      "StringEqualsIfExists" : {
        "dynamodb:Select" : "SPECIFIC_ATTRIBUTES"
      }
    }
  }
}

```

## Esempio: specificare i principi

L'esempio seguente mostra come creare una politica basata sulle risorse che neghi l'accesso a un bucket a tutti i principali ad eccezione di quelli specificati nella condizione.

```
public String specifyPrincipalsExample() {
    IamPolicy policy = IamPolicy.builder()
        .addStatement(b -> b
            .effect(IamEffect.DENY)
            .addAction("s3:*")
            .addPrincipal(IamPrincipal.ALL)
            .addResource("arn:aws:s3:::BUCKETNAME/*")
            .addResource("arn:aws:s3:::BUCKETNAME")
            .addCondition(b1 -> b1
                .operator(IamConditionOperator.ARN_NOT_EQUALS)
                .key("aws:PrincipalArn")
                .value("arn:aws:iam::444455556666:user/user-name")))
        .build();
    return policy.toJson(IamPolicyWriter.builder()
        .prettyPrint(true).build());
}
```

## Output JSON

L'ultima istruzione dell'esempio precedente restituisce la seguente stringa JSON.

Ulteriori informazioni su questo [esempio sono disponibili](#) nella Guida per l'AWS Identity and Access Management utente.

```
{
  "Version" : "2012-10-17",
  "Statement" : {
    "Effect" : "Deny",
    "Principal" : "*",
    "Action" : "s3:*",
    "Resource" : [ "arn:aws:s3:::BUCKETNAME/*", "arn:aws:s3:::BUCKETNAME" ],
    "Condition" : {
      "ArnNotEquals" : {
        "aws:PrincipalArn" : "arn:aws:iam::444455556666:user/user-name"
      }
    }
  }
}
```

## Esempio: consentire l'accesso su più account

L'esempio seguente mostra come consentire Account AWS a un altro utente di caricare oggetti nel tuo bucket mantenendo il pieno controllo del proprietario degli oggetti caricati.

```
public String allowCrossAccountAccessExample() {
    IamPolicy policy = IamPolicy.builder()
        .addStatement(b -> b
            .effect(IamEffect.ALLOW)
            .addPrincipal(IamPrincipalType.AWS, "111122223333")
            .addAction("s3:PutObject")
            .addResource("arn:aws:s3:::DOC-EXAMPLE-BUCKET/*")
            .addCondition(b1 -> b1
                .operator(IamConditionOperator.STRING_EQUALS)
                .key("s3:x-amz-acl")
                .value("bucket-owner-full-control")))
        .build();
    return policy.toJson(IamPolicyWriter.builder()
        .prettyPrint(true).build());
}
```

## Output JSON

L'ultima istruzione dell'esempio precedente restituisce la seguente stringa JSON.

Scopri di più su questo [esempio](#) nella Amazon Simple Storage Service User Guide.

```
{
  "Version" : "2012-10-17",
  "Statement" : {
    "Effect" : "Allow",
    "Principal" : {
      "AWS" : "111122223333"
    },
    "Action" : "s3:PutObject",
    "Resource" : "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*",
    "Condition" : {
      "StringEquals" : {
        "s3:x-amz-acl" : "bucket-owner-full-control"
      }
    }
  }
}
```



## Usa un **IamPolicy** con IAM

Dopo aver creato un'IamPolicyistanza, usi un [IamClient](#) per lavorare con il servizio IAM.

L'esempio seguente crea una policy che consente a un'[identità IAM](#) di scrivere elementi in una tabella DynamoDB nell'account specificato con il parametro. accountID La policy viene quindi caricata su IAM come stringa JSON.

```
public String createAndUploadPolicyExample(IamClient iam, String accountID, String
policyName) {
    // Build the policy.
    IamPolicy policy =
        IamPolicy.builder() // 'version' defaults to "2012-10-17".
            .addStatement(IamStatement.builder()
                .effect(IamEffect.ALLOW)
                .addAction("dynamodb:PutItem")
                .addResource("arn:aws:dynamodb:us-east-1:" + accountID
+ ":table/exampleTableName")
            ).build())
        .build();
    // Upload the policy.
    iam.createPolicy(r ->
r.policyName(policyName).policyDocument(policy.toJson()));
    return policy.toJson(IamPolicyWriter.builder().prettyPrint(true).build());
}
```

L'esempio successivo si basa sull'esempio precedente. Il codice scarica la policy e la utilizza come base per una nuova policy copiando e alterando la dichiarazione. La nuova politica viene quindi caricata.

```
public String createNewBasedOnExistingPolicyExample(IamClient iam, String
accountID, String policyName, String newPolicyName) {

    String policyArn = "arn:aws:iam::" + accountID + ":policy/" + policyName;
    GetPolicyResponse getPolicyResponse = iam.getPolicy(r ->
r.policyArn(policyArn));

    String policyVersion = getPolicyResponse.policy().defaultVersionId();
    GetPolicyVersionResponse getPolicyVersionResponse =
        iam.getPolicyVersion(r ->
r.policyArn(policyArn).versionId(policyVersion));
```

```

    // Create an IamPolicy instance from the JSON string returned from IAM.
    String decodedPolicy =
URLDecoder.decode(getPolicyVersionResponse.policyVersion().document(),
StandardCharsets.UTF_8);
    IamPolicy policy = IamPolicy.fromJson(decodedPolicy);

    /*
    All IamPolicy components are immutable, so use the copy method that
creates a new instance that
    can be altered in the same method call.

    Add the ability to get an item from DynamoDB as an additional action.
    */
    IamStatement newStatement = policy.statements().get(0).copy(s ->
s.addAction("dynamodb:GetItem"));

    // Create a new statement that replaces the original statement.
    IamPolicy newPolicy = policy.copy(p ->
p.statements(Arrays.asList(newStatement)));

    // Upload the new policy. IAM now has both policies.
    iam.createPolicy(r -> r.policyName(newPolicyName)
        .policyDocument(newPolicy.toJson()));

    return newPolicy.toJson(IamPolicyWriter.builder().prettyPrint(true).build());
}

```

## IamClient

Gli esempi precedenti utilizzano un `IamClient` argomento creato come illustrato nel frammento seguente.

```
IamClient iam = IamClient.builder().region(Region.AWS_GLOBAL).build();
```

## Politiche in JSON

Gli esempi restituiscono le seguenti stringhe JSON.

```

First example
{
  "Version" : "2012-10-17",
  "Statement" : {
    "Effect" : "Allow",

```

```

    "Action" : "dynamodb:PutItem",
    "Resource" : "arn:aws:dynamodb:us-east-1:111122223333:table/exampleTableName"
  }
}

```

Second example

```

{
  "Version" : "2012-10-17",
  "Statement" : {
    "Effect" : "Allow",
    "Action" : [ "dynamodb:PutItem", "dynamodb:GetItem" ],
    "Resource" : "arn:aws:dynamodb:us-east-1:111122223333:table/exampleTableName"
  }
}

```

## Lavora con IAM le politiche

### Creazione di una policy

Per creare una nuova policy, fornisci il nome della policy e un documento di policy in formato JSON utilizzando un [CreatePolicyRequest](#) metodo to the's. `IamClient createPolicy`

### Importazioni

```

import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.services.iam.model.CreatePolicyRequest;
import software.amazon.awssdk.services.iam.model.CreatePolicyResponse;
import software.amazon.awssdk.services.iam.model.GetPolicyRequest;
import software.amazon.awssdk.services.iam.model.GetPolicyResponse;
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.waiters.IamWaiter;

```

### Codice

```

public static String createIAMPolicy(IamClient iam, String policyName ) {

    try {
        // Create an IamWaiter object
        IamWaiter iamWaiter = iam.waiter();

```

```
        CreatePolicyRequest request = CreatePolicyRequest.builder()
            .policyName(policyName)
            .policyDocument(PolicyDocument).build();

        CreatePolicyResponse response = iam.createPolicy(request);

        // Wait until the policy is created
        GetPolicyRequest polRequest = GetPolicyRequest.builder()
            .policyArn(response.policy().arn())
            .build();

        WaiterResponse<GetPolicyResponse> waitUntilPolicyExists =
iamWaiter.waitUntilPolicyExists(polRequest);
        waitUntilPolicyExists.matched().response().ifPresent(System.out::println);
        return response.policy().arn();

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "" ;
}
```

[Vedi l'esempio completo su GitHub](#)

## Ottenere una policy

Per recuperare una policy esistente, chiamate il `getPolicy` metodo `IamClient`'s, fornendo l'ARN della policy all'interno [GetPolicyRequest](#) di un oggetto.

### Importazioni

```
import software.amazon.awssdk.services.iam.model.GetPolicyRequest;
import software.amazon.awssdk.services.iam.model.GetPolicyResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;
```

### Codice

```
public static void getIAMPolicy(IamClient iam, String policyArn) {
```

```
try {
    GetPolicyRequest request = GetPolicyRequest.builder()
        .policyArn(policyArn).build();

    GetPolicyResponse response = iam.getPolicy(request);
    System.out.format("Successfully retrieved policy %s",
        response.policy().policyName());

} catch (IamException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

Vedi l'[esempio completo](#) su GitHub

## Allegare una policy dei ruoli

Puoi allegare una policy a un IAM [ruolo](#) chiamando il `attachRolePolicy` metodo `IamClient` th's, fornendogli il nome del ruolo e l'ARN della policy in un [AttachRolePolicyRequest](#)

### Importazioni

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.services.iam.model.AttachRolePolicyRequest;
import software.amazon.awssdk.services.iam.model.AttachedPolicy;
import software.amazon.awssdk.services.iam.model.ListAttachedRolePoliciesRequest;
import software.amazon.awssdk.services.iam.model.ListAttachedRolePoliciesResponse;
import java.util.List;
```

### Codice

```
public static void attachIAMRolePolicy(IamClient iam, String roleName, String
policyArn ) {

    try {

        ListAttachedRolePoliciesRequest request =
ListAttachedRolePoliciesRequest.builder()
            .roleName(roleName)
```

```
        .build();

        ListAttachedRolePoliciesResponse response =
iam.listAttachedRolePolicies(request);
        List<AttachedPolicy> attachedPolicies = response.attachedPolicies();

        // Ensure that the policy is not attached to this role
        String polArn = "";
        for (AttachedPolicy policy: attachedPolicies) {
            polArn = policy.policyArn();
            if (polArn.compareTo(policyArn)==0) {
                System.out.println(roleName +
                    " policy is already attached to this role.");
                return;
            }
        }

        AttachRolePolicyRequest attachRequest =
            AttachRolePolicyRequest.builder()
                .roleName(roleName)
                .policyArn(policyArn)
                .build();

        iam.attachRolePolicy(attachRequest);

        System.out.println("Successfully attached policy " + policyArn +
            " to role " + roleName);

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }

    System.out.println("Done");
}
```

Vedi l'[esempio completo](#) su. GitHub

## Elencare le policy dei ruoli collegate

Elenca le politiche allegate su un ruolo chiamando il `listAttachedRolePolicies` metodo `IamClient`'s. È necessario un [ListAttachedRolePoliciesRequest](#) oggetto che contenga il nome del ruolo per cui elencare le politiche.

Richiama `getAttachedPolicies` l'[ListAttachedRolePoliciesResponse](#) oggetto restituito per ottenere l'elenco delle politiche allegate. I risultati possono essere troncati; se il metodo `ListAttachedRolePoliciesResponse` dell'oggetto `isTruncated` restituisce `true`, chiama il metodo `ListAttachedRolePoliciesResponse` dell'oggetto `marker`. Utilizza il contrassegno restituito per creare una nuova richiesta e utilizzala per chiamare nuovamente `listAttachedRolePolicies` per ottenere il batch di risultati successivo.

## Importazioni

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.services.iam.model.AttachRolePolicyRequest;
import software.amazon.awssdk.services.iam.model.AttachedPolicy;
import software.amazon.awssdk.services.iam.model.ListAttachedRolePoliciesRequest;
import software.amazon.awssdk.services.iam.model.ListAttachedRolePoliciesResponse;
import java.util.List;
```

## Codice

```
public static void attachIAMRolePolicy(IamClient iam, String roleName, String
policyArn ) {

    try {

        ListAttachedRolePoliciesRequest request =
ListAttachedRolePoliciesRequest.builder()
            .roleName(roleName)
            .build();

        ListAttachedRolePoliciesResponse response =
iam.listAttachedRolePolicies(request);
        List<AttachedPolicy> attachedPolicies = response.attachedPolicies();

        // Ensure that the policy is not attached to this role
        String polArn = "";
        for (AttachedPolicy policy: attachedPolicies) {
            polArn = policy.policyArn();
            if (polArn.compareTo(policyArn)==0) {
                System.out.println(roleName +
                    " policy is already attached to this role.");
                return;
            }
        }
    }
}
```

```
        }
    }

    AttachRolePolicyRequest attachRequest =
        AttachRolePolicyRequest.builder()
            .roleName(roleName)
            .policyArn(policyArn)
            .build();

    iam.attachRolePolicy(attachRequest);

    System.out.println("Successfully attached policy " + policyArn +
        " to role " + roleName);

} catch (IamException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}

System.out.println("Done");
}
```

Vedi l'[esempio completo](#) su GitHub.

## Scollegare una policy dei ruoli

Per scollegare una policy da un ruolo, chiamate il `detachRolePolicy` metodo `IamClient` th's, fornendogli il nome del ruolo e l'ARN della policy in a. [DetachRolePolicyRequest](#)

### Importazioni

```
import software.amazon.awssdk.services.iam.model.DetachRolePolicyRequest;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;
```

### Codice

```
public static void detachPolicy(IamClient iam, String roleName, String policyArn )
{

    try {
```



```
DetachRolePolicyRequest request = DetachRolePolicyRequest.builder()
    .roleName(roleName)
    .policyArn(policyArn)
    .build();

iam.detachRolePolicy(request);
System.out.println("Successfully detached policy " + policyArn +
    " from role " + roleName);

} catch (IamException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

Vedi l'[esempio completo](#) su GitHub

## Ulteriori informazioni

- [Panoramica delle IAM politiche](#) nella Guida per l'IAMutente.
- [AWSRiferimento alla politica IAM](#) nella Guida IAM per l'utente.
- [CreatePolicy](#)nell'IAMAPI Reference
- [GetPolicy](#)nell'IAMAPI Reference
- [AttachRolePolicy](#)nell'IAMAPI Reference
- [ListAttachedRolePolicies](#)nell'IAMAPI Reference
- [DetachRolePolicy](#)nell'IAMAPI Reference

## Lavora con i certificati IAM del server

Per abilitare le connessioni HTTPS al tuo sito Web o alla tua applicazione AWS, devi disporre di un certificato del server SSL/TLS. È possibile utilizzare un certificato server fornito da AWS Certificate Manager o ottenuto da un provider esterno.

Si consiglia di ACM utilizzarlo per fornire, gestire e distribuire i certificati del server. Con ACM puoi richiedere un certificato, distribuirlo nelle tue AWS risorse e lasciare che ACM gestisca i rinnovi dei certificati al posto tuo. I certificati forniti da ACM sono gratuiti. Per ulteriori informazioni in merito ACM, consulta la [Guida AWS Certificate Manager per l'utente](#).

## Recupero di un certificato del server

È possibile recuperare un certificato del server chiamando il `getServerCertificate` metodo `IamClient`'s e passandolo [GetServerCertificateRequest](#) con il nome del certificato.

### Importazioni

```
import software.amazon.awssdk.services.iam.model.GetServerCertificateRequest;
import software.amazon.awssdk.services.iam.model.GetServerCertificateResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;
```

### Codice

```
public static void getCertificate(IamClient iam,String certName ) {

    try {
        GetServerCertificateRequest request = GetServerCertificateRequest.builder()
            .serverCertificateName(certName)
            .build();

        GetServerCertificateResponse response = iam.getServerCertificate(request);
        System.out.format("Successfully retrieved certificate with body %s",
            response.getServerCertificate().certificateBody());

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

Vedi l'[esempio completo](#) su GitHub

## Elencare i certificati del server

Per elencare i certificati del tuo server, chiama il `listServerCertificates` metodo `IamClient`'s con un [ListServerCertificatesRequest](#). Restituisce [ListServerCertificatesResponse](#).

Chiama il `serverCertificateMetadataList` metodo dell'`ListServerCertificateResponse` oggetto restituito per ottenere un elenco di [ServerCertificateMetadata](#) oggetti che puoi utilizzare per ottenere informazioni su ciascun certificato.

I risultati possono essere troncati; se il metodo `ListServerCertificateResponse` dell'oggetto `isTruncated` restituisce `true`, chiamare il metodo `ListServerCertificatesResponse` dell'oggetto `marker` e utilizzare il contrassegno per creare una nuova richiesta. Utilizza la nuova richiesta per chiamare nuovamente `listServerCertificates` per ottenere il batch di risultati successivo.

## Importazioni

```
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.services.iam.model.ListServerCertificatesRequest;
import software.amazon.awssdk.services.iam.model.ListServerCertificatesResponse;
import software.amazon.awssdk.services.iam.model.ServerCertificateMetadata;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
```

## Codice

```
public static void listCertificates(IamClient iam) {

    try {
        boolean done = false;
        String newMarker = null;

        while(!done) {
            ListServerCertificatesResponse response;

            if (newMarker == null) {
                ListServerCertificatesRequest request =
                    ListServerCertificatesRequest.builder().build();
                response = iam.listServerCertificates(request);
            } else {
                ListServerCertificatesRequest request =
                    ListServerCertificatesRequest.builder()
                        .marker(newMarker).build();
                response = iam.listServerCertificates(request);
            }

            for(ServerCertificateMetadata metadata :
                response.serverCertificateMetadataList()) {
                System.out.printf("Retrieved server certificate %s",
                    metadata.serverCertificateName());
            }
        }
    }
}
```

```
        if(!response.isTruncated()) {
            done = true;
        } else {
            newMarker = response.marker();
        }
    }

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

Vedi l'[esempio completo](#) su GitHub.

## Aggiornamento di un certificato del server

È possibile aggiornare il nome o il percorso di un certificato del server chiamando il `updateServerCertificate` metodo `IamClient`'s. Richiede un [UpdateServerCertificateRequest](#) oggetto impostato con il nome corrente del certificato del server e un nuovo nome o un nuovo percorso da utilizzare.

### Importazioni

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.services.iam.model.UpdateServerCertificateRequest;
import software.amazon.awssdk.services.iam.model.UpdateServerCertificateResponse;
```

### Codice

```
public static void updateCertificate(IamClient iam, String curName, String newName)
{
    try {
        UpdateServerCertificateRequest request =
            UpdateServerCertificateRequest.builder()
                .serverCertificateName(curName)
                .newServerCertificateName(newName)
                .build();
```

```
UpdateServerCertificateResponse response =
    iam.updateServerCertificate(request);

System.out.printf("Successfully updated server certificate to name %s",
    newName);

} catch (IamException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

Vedi l'[esempio completo](#) su GitHub.

## Eliminazione di un certificato del server

Per eliminare un certificato del server, chiama il `deleteServerCertificate` metodo `IamClient`'s con un [DeleteServerCertificateRequest](#) contenente il nome del certificato.

### Importazioni

```
import software.amazon.awssdk.services.iam.model.DeleteServerCertificateRequest;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;
```

### Codice

```
public static void deleteCert(IamClient iam,String certName ) {

    try {
        DeleteServerCertificateRequest request =
            DeleteServerCertificateRequest.builder()
                .serverCertificateName(certName)
                .build();

        iam.deleteServerCertificate(request);
        System.out.println("Successfully deleted server certificate " +
            certName);
    }
}
```

```
    } catch (IamException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

Vedi l'[esempio completo](#) su GitHub.

## Ulteriori informazioni

- [Utilizzo dei certificati server](#) nella Guida per l' IAM utente
- [GetServerCertificates](#) nell' IAM API Reference
- [ListServerCertificates](#) nell' IAM API Reference
- [UpdateServerCertificate](#) nell' IAM API Reference
- [DeleteServerCertificate](#) nell' IAM API Reference
- [AWS Certificate Manager Guida per l'utente](#)

## Lavora con Kinesis

Questa sezione fornisce esempi di programmazione [Amazon Kinesis](#) con AWS SDK for Java 2.x.

Per ulteriori informazioni in merito Kinesis, consulta la [Guida per gli Amazon Kinesis sviluppatori](#).

Gli esempi seguenti includono solo il codice necessario per dimostrare ciascuna tecnica. Il [codice di esempio completo è disponibile su GitHub](#). Da qui puoi scaricare un singolo file sorgente o clonare l'archivio localmente per ottenere tutti gli esempi da creare ed eseguire.

### Argomenti

- [Sottoscrizione ad Amazon Kinesis Data Streams](#)

## Sottoscrizione ad Amazon Kinesis Data Streams

Gli esempi seguenti mostrano come recuperare ed elaborare i Amazon Kinesis dati da Data Streams utilizzando il metodo. `subscribeToShard` Kinesis Data Streams ora utilizza la funzionalità fanout avanzata e un'API di recupero dati HTTP/2 a bassa latenza, che facilita agli sviluppatori l'esecuzione di più applicazioni a bassa latenza e ad alte prestazioni sullo stesso Data Stream. Kinesis

## Configurazione

Innanzitutto, [SubscribeToShardRequest](#) crea un client asincrono e un oggetto. Kinesis Questi oggetti vengono utilizzati in ciascuno dei seguenti esempi per eseguire la sottoscrizione agli eventi Kinesis.

### Importazioni

```
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.atomic.AtomicInteger;
import org.reactivestreams.Subscriber;
import org.reactivestreams.Subscription;
import software.amazon.awssdk.core.async.SdkPublisher;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.kinesis.KinesisAsyncClient;
import software.amazon.awssdk.services.kinesis.model.ShardIteratorType;
import software.amazon.awssdk.services.kinesis.model.SubscribeToShardEvent;
import software.amazon.awssdk.services.kinesis.model.SubscribeToShardEventStream;
import software.amazon.awssdk.services.kinesis.model.SubscribeToShardRequest;
import software.amazon.awssdk.services.kinesis.model.SubscribeToShardResponse;
import software.amazon.awssdk.services.kinesis.model.SubscribeToShardResponseHandler;
```

### Codice

```
Region region = Region.US_EAST_1;
KinesisAsyncClient client = KinesisAsyncClient.builder()
    .region(region)
    .build();

SubscribeToShardRequest request = SubscribeToShardRequest.builder()
    .consumerARN(CONSUMER_ARN)
    .shardId("arn:aws:kinesis:us-east-1:111122223333:stream/
StockTradeStream")
    .startingPosition(s -> s.type(ShardIteratorType.LATEST)).build();
```

## Usa l'interfaccia del generatore

È possibile utilizzare il `builder` metodo per semplificare la creazione di.

### [SubscribeToShardResponseHandler](#)

Utilizzando il generatore, è possibile impostare la richiamata del ciclo di vita con una chiamata al metodo invece che implementando l'interfaccia completa.

## Codice

```
private static CompletableFuture<Void> responseHandlerBuilder(KinesisAsyncClient
client, SubscribeToShardRequest request) {
    SubscribeToShardResponseHandler responseHandler =
SubscribeToShardResponseHandler
    .builder()
    .onError(t -> System.err.println("Error during stream - " +
t.getMessage()))
    .onComplete(() -> System.out.println("All records stream
successfully"))
    // Must supply some type of subscriber
    .subscriber(e -> System.out.println("Received event - " + e))
    .build();
    return client.subscribeToShard(request, responseHandler);
}
```

Per un maggiore controllo del publisher, è possibile utilizzare il metodo `publisherTransformer` per personalizzare il publisher.

## Codice

```
private static CompletableFuture<Void>
responseHandlerBuilderPublisherTransformer(KinesisAsyncClient client,
SubscribeToShardRequest request) {
    SubscribeToShardResponseHandler responseHandler =
SubscribeToShardResponseHandler
    .builder()
    .onError(t -> System.err.println("Error during stream - " +
t.getMessage()))
    .publisherTransformer(p -> p.filter(e -> e instanceof
SubscribeToShardEvent).limit(100))
    .subscriber(e -> System.out.println("Received event - " + e))
    .build();
    return client.subscribeToShard(request, responseHandler);
}
```

Vedi l'[esempio completo](#) su GitHub.

## Usa un gestore di risposte personalizzato

Per il pieno controllo dell'abbonato e dell'editore, implementa l'`SubscribeToShardResponseHandler` interfaccia.



In questo esempio implementi il metodo `onEventStream`, che ti consente l'accesso completo al publisher. Questo dimostra come trasformare il publisher in record di eventi da stampare da parte del sottoscrittore.

## Codice

```
private static CompletableFuture<Void>
responseHandlerBuilderClassic(KinesisAsyncClient client, SubscribeToShardRequest
request) {
    SubscribeToShardResponseHandler responseHandler = new
SubscribeToShardResponseHandler() {

        @Override
        public void responseReceived(SubscribeToShardResponse response) {
            System.out.println("Receieved initial response");
        }

        @Override
        public void onEventStream(SdkPublisher<SubscribeToShardEventStream>
publisher) {
            publisher
                // Filter to only SubscribeToShardEvents
                .filter(SubscribeToShardEvent.class)
                // Flat map into a publisher of just records
                .flatMapIterable(SubscribeToShardEvent::records)
                // Limit to 1000 total records
                .limit(1000)
                // Batch records into lists of 25
                .buffer(25)
                // Print out each record batch
                .subscribe(batch -> System.out.println("Record Batch - " +
batch));
        }

        @Override
        public void complete() {
            System.out.println("All records stream successfully");
        }

        @Override
        public void exceptionOccurred(Throwable throwable) {
            System.err.println("Error during stream - " + throwable.getMessage());
        }
    }
}
```

```
};
return client.subscribeToShard(request, responseHandler);
}
```

Vedi l'[esempio completo](#) su GitHub

## Usa l'interfaccia per i visitatori

È possibile utilizzare un oggetto [Visitatore](#) per iscriverti a eventi specifici che ti interessa guardare.

### Codice

```
private static CompletableFuture<Void>
responseHandlerBuilderVisitorBuilder(KinesisAsyncClient client,
SubscribeToShardRequest request) {
    SubscribeToShardResponseHandler.Visitor visitor =
SubscribeToShardResponseHandler.Visitor
        .builder()
        .onSubscribeToShardEvent(e -> System.out.println("Received subscribe to
shard event " + e))
        .build();
    SubscribeToShardResponseHandler responseHandler =
SubscribeToShardResponseHandler
        .builder()
        .onError(t -> System.err.println("Error during stream - " +
t.getMessage()))
        .subscriber(visitor)
        .build();
    return client.subscribeToShard(request, responseHandler);
}
```

Guarda l'[esempio completo](#) su GitHub.

## Usa un abbonato personalizzato

Puoi anche implementare i tuoi sottoscrittori personalizzati per iscriverti al flusso.

Questo frammento di codice mostra un esempio di sottoscrittore.

### Codice

```
private static class MySubscriber implements
Subscriber<SubscribeToShardEventStream> {
```

```

private Subscription subscription;
private AtomicInteger eventCount = new AtomicInteger(0);

@Override
public void onSubscribe(Subscription subscription) {
    this.subscription = subscription;
    this.subscription.request(1);
}

@Override
public void onNext(SubscribeToShardEventStream shardSubscriptionEventStream) {
    System.out.println("Received event " + shardSubscriptionEventStream);
    if (eventCount.incrementAndGet() >= 100) {
        // You can cancel the subscription at any time if you wish to stop
receiving events.
        subscription.cancel();
    }
    subscription.request(1);
}

@Override
public void onError(Throwable throwable) {
    System.err.println("Error occurred while stream - " +
throwable.getMessage());
}

@Override
public void onComplete() {
    System.out.println("Finished streaming all events");
}
}

```

È possibile passare l'abbonato personalizzato al `subscribe` metodo come illustrato nel seguente frammento di codice.

## Codice

```

private static CompletableFuture<Void>
responseHandlerBuilderSubscriber(KinesisAsyncClient client, SubscribeToShardRequest
request) {
    SubscribeToShardResponseHandler responseHandler =
SubscribeToShardResponseHandler
        .builder()

```

```

        .onError(t -> System.err.println("Error during stream - " +
t.getMessage()))
        .subscriber(MySubscriber::new)
        .build();
    return client.subscribeToShard(request, responseHandler);
}

```

Vedi l'esempio [completo](#) su GitHub

## Scrivi i record di dati in un flusso di Kinesis dati

È possibile utilizzare l'[KinesisClient](#) oggetto per scrivere record di dati in un flusso di Kinesis dati utilizzando il `putRecords` metodo. Per richiamare correttamente questo metodo, create un [PutRecordsRequest](#) oggetto. Si passa il nome del flusso di dati al `streamName` metodo. Inoltre devi passare i dati utilizzando il metodo `putRecords` (come mostrato nell'esempio di codice seguente).

### Importazioni

```

import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.kinesis.KinesisClient;
import software.amazon.awssdk.services.kinesis.model.PutRecordRequest;
import software.amazon.awssdk.services.kinesis.model.KinesisException;
import software.amazon.awssdk.services.kinesis.model.DescribeStreamRequest;
import software.amazon.awssdk.services.kinesis.model.DescribeStreamResponse;

```

Nel seguente esempio di codice Java, notate che l'`StockTrade` oggetto viene utilizzato come dati da scrivere nel flusso di Kinesis dati. Prima di eseguire questo esempio, assicuratevi di aver creato il flusso di dati.

### Codice

```

import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.kinesis.KinesisClient;
import software.amazon.awssdk.services.kinesis.model.PutRecordRequest;
import software.amazon.awssdk.services.kinesis.model.KinesisException;
import software.amazon.awssdk.services.kinesis.model.DescribeStreamRequest;
import software.amazon.awssdk.services.kinesis.model.DescribeStreamResponse;

/**
 * Before running this Java V2 code example, set up your development

```

```
* environment, including your credentials.
*
* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class StockTradesWriter {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <streamName>

            Where:
                streamName - The Amazon Kinesis data stream to which records are
written (for example, StockTradeStream)
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String streamName = args[0];
        Region region = Region.US_EAST_1;
        KinesisClient kinesisClient = KinesisClient.builder()
            .region(region)
            .build();

        // Ensure that the Kinesis Stream is valid.
        validateStream(kinesisClient, streamName);
        setStockData(kinesisClient, streamName);
        kinesisClient.close();
    }

    public static void setStockData(KinesisClient kinesisClient, String streamName) {
        try {
            // Repeatedly send stock trades with a 100 milliseconds wait in between.
            StockTradeGenerator stockTradeGenerator = new StockTradeGenerator();

            // Put in 50 Records for this example.
            int index = 50;
            for (int x = 0; x < index; x++) {
                StockTrade trade = stockTradeGenerator.getRandomTrade();
            }
        }
    }
}
```

```
        sendStockTrade(trade, kinesisClient, streamName);
        Thread.sleep(100);
    }

    } catch (KinesisException | InterruptedException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    System.out.println("Done");
}

private static void sendStockTrade(StockTrade trade, KinesisClient kinesisClient,
    String streamName) {
    byte[] bytes = trade.toJsonAsBytes();

    // The bytes could be null if there is an issue with the JSON serialization by
    // the Jackson JSON library.
    if (bytes == null) {
        System.out.println("Could not get JSON bytes for stock trade");
        return;
    }

    System.out.println("Putting trade: " + trade);
    PutRecordRequest request = PutRecordRequest.builder()
        .partitionKey(trade.getTickerSymbol()) // We use the ticker symbol as
the partition key, explained in
// the Supplemental Information
section below.
        .streamName(streamName)
        .data(SdkBytes.fromByteArray(bytes))
        .build();

    try {
        kinesisClient.putRecord(request);
    } catch (KinesisException e) {
        System.err.println(e.getMessage());
    }
}

private static void validateStream(KinesisClient kinesisClient, String streamName)
{
    try {
        DescribeStreamRequest describeStreamRequest =
DescribeStreamRequest.builder()
```

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

        DescribeStreamResponse describeStreamResponse =
kinesisClient.describeStream(describeStreamRequest);

        if (!
describeStreamResponse.streamDescription().streamStatus().toString().equals("ACTIVE"))
        {
            System.err.println("Stream " + streamName + " is not active. Please
wait a few moments and try again.");
            System.exit(1);
        }

    } catch (KinesisException e) {
        System.err.println("Error found while describing the stream " +
streamName);
        System.err.println(e);
        System.exit(1);
    }
}
}
```

Vedi l'[esempio completo](#) su GitHub.

## Usa una libreria di terze parti

Puoi utilizzare altre librerie di terze parti anziché implementare un sottoscrittore personalizzato. Questo esempio dimostra l'utilizzo dell' RxJava implementazione, ma è possibile utilizzare qualsiasi libreria che implementa le interfacce Reactive Streams. Vedi la [pagina RxJava wiki su Github per ulteriori informazioni su](#) quella libreria.

Per utilizzare la libreria, aggiungerla come una dipendenza. Se stai utilizzando Maven, l'esempio illustra lo snippet POM da utilizzare.

### Voce POM

```
<dependency>
  <groupId>io.reactivex.rxjava2</groupId>
  <artifactId>rxjava</artifactId>
  <version>2.1.14</version>
</dependency>
```

## Importazioni

```
import java.net.URI;
import java.util.concurrent.CompletableFuture;

import io.reactivex.Flowable;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.core.async.SdkPublisher;
import software.amazon.awssdk.http.Protocol;
import software.amazon.awssdk.http.SdkHttpConfigurationOption;
import software.amazon.awssdk.http.nio.netty.NettyNioAsyncHttpClient;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.kinesis.KinesisAsyncClient;
import software.amazon.awssdk.services.kinesis.model.ShardIteratorType;
import software.amazon.awssdk.services.kinesis.model.StartingPosition;
import software.amazon.awssdk.services.kinesis.model.SubscribeToShardEvent;
import software.amazon.awssdk.services.kinesis.model.SubscribeToShardRequest;
import software.amazon.awssdk.services.kinesis.model.SubscribeToShardResponseHandler;
import software.amazon.awssdk.utils.AttributeMap;
```

Questo esempio utilizza RxJava il metodo `onEventStream lifecycle`. In questo modo hai pieno accesso al publisher, che può essere utilizzato per creare un Rx Flowable.

## Codice

```
SubscribeToShardResponseHandler responseHandler =
SubscribeToShardResponseHandler
    .builder()
    .onError(t -> System.err.println("Error during stream - " +
t.getMessage()))
    .onEventStream(p -> Flowable.fromPublisher(p)
        .ofType(SubscribeToShardEvent.class)

.flatMapIterable(SubscribeToShardEvent::records)
        .limit(1000)
        .buffer(25)
        .subscribe(e -> System.out.println("Record
batch = " + e)))
    .build();
```

Puoi anche utilizzare il metodo `publisherTransformer` con il publisher Flowable. È necessario adattare l'Flowable editore a un SdkPublisher, come illustrato nell'esempio seguente.



## Codice

```
SubscribeToShardResponseHandler responseHandler =
SubscribeToShardResponseHandler
    .builder()
    .onError(t -> System.err.println("Error during stream - " +
t.getMessage()))
    .publisherTransformer(p ->
SdkPublisher.adapt(Flowable.fromPublisher(p).limit(100)))
    .build();
```

Vedi l'[esempio completo](#) su GitHub.

## Ulteriori informazioni

- [SubscribeToShardEvent](#) nell'Amazon KinesisAPI Reference
- [SubscribeToShard](#) nell'Amazon KinesisAPI Reference

## Funzioni di richiamo, elenco ed eliminazione AWS Lambda

Questa sezione fornisce esempi di programmazione con il client di Lambda servizio utilizzando AWS SDK for Java 2.x.

### Argomenti

- [Invocazione di una funzione Lambda.](#)
- [Elenca le funzioni Lambda](#)
- [Eliminazione di una funzione Lambda](#)

## Invocazione di una funzione Lambda.

È possibile richiamare una Lambda funzione creando un [LambdaClient](#) oggetto e richiamando il relativo metodo. `invoke` Create un [InvokeRequest](#) oggetto per specificare informazioni aggiuntive come il nome della funzione e il payload da passare alla funzione. Lambda I nomi delle funzioni vengono visualizzati come `arn:aws:lambda:us-east-1:123456789012:function:. HelloFunction` È possibile recuperare il valore osservando la funzione in. AWS Management Console

Per passare i dati del payload a una funzione, create un [SdkBytes](#) oggetto che contenga informazioni. Nell'esempio di codice riportato di seguito, notare i dati JSON passati alla funzione Lambda .

## Importazioni

```
import software.amazon.awssdk.services.lambda.LambdaClient;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.lambda.model.InvokeRequest;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.services.lambda.model.InvokeResponse;
import software.amazon.awssdk.services.lambda.model.LambdaException;
```

## Codice

Il seguente esempio di codice mostra come richiamare una funzione. Lambda

```
public static void invokeFunction(LambdaClient awsLambda, String functionName) {

    InvokeResponse res = null ;
    try {
        //Need a SdkBytes instance for the payload
        String json = "{\"Hello \": \"Paris\"}";
        SdkBytes payload = SdkBytes.fromUtf8String(json) ;

        //Setup an InvokeRequest
        InvokeRequest request = InvokeRequest.builder()
            .functionName(functionName)
            .payload(payload)
            .build();

        res = awsLambda.invoke(request);
        String value = res.payload().asUtf8String() ;
        System.out.println(value);

    } catch(LambdaException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

Vedi l'[esempio completo](#) su. GitHub

## Elenca le funzioni Lambda

Costruisci un [Lambda Client](#) oggetto e invoca il suo `listFunctions` metodo. Questo metodo restituisce un [ListFunctionsResponse](#) oggetto. È possibile richiamare il `functions` metodo di questo oggetto per restituire un elenco di [FunctionConfiguration](#) oggetti. È possibile scorrere l'elenco per recuperare informazioni sulle funzioni. Il seguente esempio di codice Java mostra come ottenere i nomi di ogni funzione.

### Importazioni

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.lambda.LambdaClient;
import software.amazon.awssdk.services.lambda.model.LambdaException;
import software.amazon.awssdk.services.lambda.model.ListFunctionsResponse;
import software.amazon.awssdk.services.lambda.model.FunctionConfiguration;
import java.util.List;
```

### Codice

Nell'esempio di codice Java riportato di seguito viene illustrato come recuperare un elenco di nomi di funzioni.

```
public static void listFunctions(LambdaClient awsLambda) {

    try {
        ListFunctionsResponse functionResult = awsLambda.listFunctions();
        List<FunctionConfiguration> list = functionResult.functions();

        for (FunctionConfiguration config: list) {
            System.out.println("The function name is "+config.functionName());
        }

    } catch (LambdaException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

Vedi l'[esempio completo](#) su. GitHub

## Eliminazione di una funzione Lambda

Costruisci un [LambdaClient](#) oggetto e invoca il suo `deleteFunction` metodo. Crea un [DeleteFunctionRequest](#) oggetto e passalo al `deleteFunction` metodo. Questo oggetto contiene informazioni quali il nome della funzione da eliminare. I nomi delle funzioni vengono visualizzati come `arn:aws:lambda:us-east-1:123456789012:function:. HelloFunction`. È possibile recuperare il valore osservando la funzione in AWS Management Console

### Importazioni

```
import software.amazon.awssdk.services.lambda.LambdaClient;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.lambda.model.DeleteFunctionRequest;
import software.amazon.awssdk.services.lambda.model.LambdaException;
```

### Codice

Il seguente codice Java mostra come eliminare una Lambda funzione.

```
public static void deleteLambdaFunction(LambdaClient awsLambda, String
functionName ) {
    try {
        DeleteFunctionRequest request = DeleteFunctionRequest.builder()
            .functionName(functionName)
            .build();

        awsLambda.deleteFunction(request);
        System.out.println("The "+functionName +" function was deleted");

    } catch(LambdaException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

Vedi l'[esempio completo](#) su GitHub

## Funziona con Amazon S3

Questa sezione fornisce esempi di programmazione con [Amazon Simple Storage Service \(S3\) utilizzando](#). AWS SDK for Java 2.x

Gli esempi seguenti includono solo il codice necessario per dimostrare ciascuna tecnica. Il [codice di esempio completo è disponibile su GitHub](#). Da qui puoi scaricare un singolo file sorgente o clonare l'archivio localmente per ottenere tutti gli esempi da creare ed eseguire.

### Note

A partire dalla versione 2.18.x, AWS SDK for Java 2.x utilizza l'indirizzamento in stile host virtuale quando include un endpoint override. Questo vale purché il nome del bucket sia un'etichetta DNS valida.

Chiama il `forcePathStyle` metodo con `true` nel tuo client builder per forzare il client a utilizzare l'indirizzamento in stile path per i bucket.

L'esempio seguente mostra un client di servizio configurato con un endpoint override e che utilizza un indirizzamento in stile path.

```
S3Client client = S3Client.builder()
    .region(Region.US_WEST_2)
    .endpointOverride(URI.create("https://s3.us-
west-2.amazonaws.com"))
    .forcePathStyle(true)
    .build();
```

## Utilizza punti di accesso o punti di accesso multiregionali

Dopo aver configurato i punti di [accesso o i punti di accesso multiregionali di Amazon S3](#), puoi chiamare metodi oggetto, come `putObject` e `getObject` e fornire l'identificatore del punto di accesso anziché il nome del bucket.

Ad esempio, se l'identificatore ARN di un punto di accesso è `arn:aws:s3:us-west-2:123456789012:accesspoint/test`, è possibile utilizzare il seguente frammento per chiamare il metodo `putObject`

```
Path path = Paths.get(URI.create("file:///temp/file.txt"));

s3Client.putObject(builder -> builder
    .key("myKey")
    .bucket("arn:aws:s3:us-west-2:123456789012:accesspoint/test")
    , path);
```

Al posto della stringa ARN, puoi anche utilizzare l'[alias in stile bucket](#) del punto di accesso per il parametro. bucket

Per utilizzare Multi-Region Access Point, sostituire il bucket parametro con il Multi-Region Access Point ARN che ha il seguente formato.

```
arn:aws:s3::account-id:accesspoint/MultiRegionAccessPoint_alias
```

Aggiungi la seguente dipendenza Maven per lavorare con punti di accesso multiregionali utilizzando l'SDK for Java. [Cerca in maven central la versione più recente.](#)

```
<dependency>
  <groupId>software.amazon.awssdk</groupId>
  <artifactId>auth-crt</artifactId>
  <version>VERSION</version>
</dependency>
```

## Argomenti

- [Creare, elencare ed eliminare i Amazon S3 bucket](#)
- [Lavorare con Amazon S3 oggetti](#)
- [Lavora con URL Amazon S3 prefirmati](#)
- [Accesso interregionale per Amazon S3](#)
- [Usa un client S3 performante: client S3 basato su AWS CRT](#)
- [Trasferisci file e directory con Amazon S3 Transfer Manager](#)

## Creare, elencare ed eliminare i Amazon S3 bucket

Ogni oggetto (file) in Amazon S3 deve risiedere all'interno di un bucket. Un bucket rappresenta una raccolta (container) di oggetti. Ogni bucket deve disporre di una chiave univoca (nome). Per informazioni dettagliate sui bucket e sulla loro configurazione, consulta [Working with Amazon S3 Buckets](#) nella Guida per l'utente. Amazon Simple Storage Service

### Note

Best practice

Ti consigliamo di abilitare la regola del [AbortIncompleteMultipartUpload](#) ciclo di vita sui tuoi bucket. Amazon S3

Questa regola indica a Amazon S3 di interrompere l'esecuzione di caricamenti in più parti che non sono stati completati entro un determinato numero di giorni dopo l'avvio. Quando questo limite di tempo impostato viene superato, Amazon S3 interrompe il caricamento ed elimina i dati di caricamento incompleti.

Per ulteriori informazioni, consulta [Lifecycle Configuration for a Bucket with Versioning nella Guida per l'utente](#). Amazon Simple Storage Service

### Note

Questi frammenti di codice presuppongono che tu abbia compreso il materiale di base e che tu abbia configurato le credenziali predefinite utilizzando le informazioni in AWS [the section called "Configurazione per l'accesso Single Sign-On per l'SDK"](#)

## Creazione di un bucket

Crea un file [CreateBucketRequest](#) fornisci un nome per il bucket. Passalo al metodo di S3Client. `createBucket` Utilizza S3Client per eseguire operazioni aggiuntive, ad esempio l'elencazione o l'eliminazione di bucket come mostrato negli esempi successivi.

### Importazioni

```
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.CreateBucketRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.HeadBucketRequest;
import software.amazon.awssdk.services.s3.model.HeadBucketResponse;
import software.amazon.awssdk.services.s3.model.ListBucketsRequest;
import software.amazon.awssdk.services.s3.model.ListBucketsResponse;
import software.amazon.awssdk.services.s3.model.DeleteBucketRequest;
import software.amazon.awssdk.services.s3.waiters.S3Waiter;
```

### Codice

Per prima cosa crea un S3Client.

```
Region region = Region.US_EAST_1;
S3Client s3 = S3Client.builder()
    .region(region)
    .build();
```

## Effettua una richiesta Crea Bucket.

```
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.CreateBucketRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.HeadBucketRequest;
import software.amazon.awssdk.services.s3.model.HeadBucketResponse;
import software.amazon.awssdk.services.s3.model.ListBucketsRequest;
import software.amazon.awssdk.services.s3.model.ListBucketsResponse;
import software.amazon.awssdk.services.s3.model.DeleteBucketRequest;
import software.amazon.awssdk.services.s3.waiters.S3Waiter;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class S3BucketOps {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        String bucket = "bucket" + System.currentTimeMillis();
        System.out.println(bucket);
        createBucket(s3, bucket);
        performOperations(s3, bucket);
    }

    // Create a bucket by using a S3Waiter object
    public static void createBucket(S3Client s3Client, String bucketName) {
```



```
try {
    S3Waiter s3Waiter = s3Client.waiter();
    CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
        .bucket(bucketName)
        .build();

    s3Client.createBucket(bucketRequest);
    HeadBucketRequest bucketRequestWait = HeadBucketRequest.builder()
        .bucket(bucketName)
        .build();

    // Wait until the bucket is created and print out the response.
    WaiterResponse<HeadBucketResponse> waiterResponse =
s3Waiter.waitUntilBucketExists(bucketRequestWait);
    waiterResponse.matched().response().ifPresent(System.out::println);
    System.out.println(bucketName + " is ready");

} catch (S3Exception e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

Vedi l'[esempio completo](#) su GitHub

## Elenco di bucket

Costruisci un [ListBucketsRequest](#). Usa il `listBuckets` metodo di `S3Client` per recuperare l'elenco dei bucket. Se la richiesta ha esito positivo, viene restituito un [ListBucketsResponse](#). Utilizza questo oggetto di risposta per recuperare l'elenco dei bucket.

## Importazioni

```
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.CreateBucketRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.HeadBucketRequest;
import software.amazon.awssdk.services.s3.model.HeadBucketResponse;
import software.amazon.awssdk.services.s3.model.ListBucketsRequest;
import software.amazon.awssdk.services.s3.model.ListBucketsResponse;
import software.amazon.awssdk.services.s3.model.DeleteBucketRequest;
```

```
import software.amazon.awssdk.services.s3.waiters.S3Waiter;
```

## Codice

Per prima cosa crea un S3Client.

```
Region region = Region.US_EAST_1;
S3Client s3 = S3Client.builder()
    .region(region)
    .build();
```

Effettua una richiesta Elenca bucket.

```
// List buckets
ListBucketsRequest listBucketsRequest = ListBucketsRequest.builder().build();
ListBucketsResponse listBucketsResponse = s3.listBuckets(listBucketsRequest);
listBucketsResponse.buckets().stream().forEach(x ->
System.out.println(x.name()));
```

Vedi l'[esempio completo](#) su GitHub

## Eliminazione di un bucket

Prima di eliminare un bucket Amazon S3, occorre accertarsi che il bucket sia vuoto o che il servizio restituirà un errore. Se si dispone di un [bucket con versione](#), occorre eliminare anche gli eventuali oggetti con versione che si trovano nel bucket.

### Argomenti

- [Eliminare oggetti in un bucket](#)
- [Eliminazione di un bucket vuoto](#)

### Eliminare oggetti in un bucket

Crea una [ListObjectsV2Request](#) e usa il `listObjects` metodo di `S3Client` per recuperare l'elenco degli oggetti nel bucket. Quindi utilizza il metodo `deleteObject` su ogni oggetto per eliminarlo.

### Importazioni

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
```

```
import software.amazon.awssdk.services.s3.model.DeleteBucketRequest;
import software.amazon.awssdk.services.s3.model.DeleteObjectRequest;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Request;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Response;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.S3Object;
```

## Codice

Per prima cosa crea un S3Client.

```
ProfileCredentialsProvider credentialsProvider =
ProfileCredentialsProvider.create();
Region region = Region.US_EAST_1;
S3Client s3 = S3Client.builder()
    .region(region)
    .credentialsProvider(credentialsProvider)
    .build();
```

Elimina tutti gli oggetti nel bucket.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.DeleteBucketRequest;
import software.amazon.awssdk.services.s3.model.DeleteObjectRequest;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Request;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Response;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.S3Object;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class S3BucketDeletion {
    public static void main(String[] args) throws Exception {
        final String usage = ""
```

Usage:

```
<bucket>
```

Where:

```
    bucket - The bucket to delete (for example, bucket1).\s
    """;
```

```
if (args.length != 1) {
    System.out.println(usage);
    System.exit(1);
}
```

```
String bucket = args[0];
Region region = Region.US_EAST_1;
S3Client s3 = S3Client.builder()
    .region(region)
    .build();
```

```
deleteObjectsInBucket(s3, bucket);
s3.close();
```

```
}
```

```
public static void deleteObjectsInBucket(S3Client s3, String bucket) {
    try {
        // To delete a bucket, all the objects in the bucket must be deleted first.
        ListObjectsV2Request listObjectsV2Request = ListObjectsV2Request.builder()
            .bucket(bucket)
            .build();
        ListObjectsV2Response listObjectsV2Response;

        do {
            listObjectsV2Response = s3.listObjectsV2(listObjectsV2Request);
            for (S3Object s3Object : listObjectsV2Response.contents()) {
                DeleteObjectRequest request = DeleteObjectRequest.builder()
                    .bucket(bucket)
                    .key(s3Object.key())
                    .build();
                s3.deleteObject(request);
            }
        } while (listObjectsV2Response.isTruncated());
        DeleteBucketRequest deleteBucketRequest =
DeleteBucketRequest.builder().bucket(bucket).build();
        s3.deleteBucket(deleteBucketRequest);

    } catch (S3Exception e) {
```

```
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

Vedi l'[esempio completo](#) su GitHub

## Eliminazione di un bucket vuoto

Crea un file [DeleteBucketRequest](#) con un nome da bucket e passalo al metodo di S3Client.

deleteBucket

## Importazioni

```
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.CreateBucketRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.HeadBucketRequest;
import software.amazon.awssdk.services.s3.model.HeadBucketResponse;
import software.amazon.awssdk.services.s3.model.ListBucketsRequest;
import software.amazon.awssdk.services.s3.model.ListBucketsResponse;
import software.amazon.awssdk.services.s3.model.DeleteBucketRequest;
import software.amazon.awssdk.services.s3.waiters.S3Waiter;
```

## Codice

Per prima cosa crea un S3Client.

```
Region region = Region.US_EAST_1;
S3Client s3 = S3Client.builder()
    .region(region)
    .build();
```

Elimina il bucket.

```
DeleteBucketRequest deleteBucketRequest = DeleteBucketRequest.builder()
    .bucket(bucket)
```

```
.build();  
  
s3.deleteBucket(deleteBucketRequest);  
s3.close();
```

Vedi l'[esempio completo](#) su GitHub

## Lavorare con Amazon S3 oggetti

Un oggetto Amazon S3 rappresenta un file o una raccolta di dati. Ogni oggetto deve essere contenuto in un [bucket](#).

### Note

#### Best practice

Ti consigliamo di abilitare la regola del [AbortIncompleteMultipartUpload](#) ciclo di vita nei tuoi bucket. Amazon S3

Questa regola indica a Amazon S3 di interrompere l'esecuzione di caricamenti in più parti che non sono stati completati entro un determinato numero di giorni dopo l'avvio. Quando questo limite di tempo impostato viene superato, Amazon S3 interrompe il caricamento ed elimina i dati di caricamento incompleti.

Per ulteriori informazioni, consulta [Lifecycle Configuration for a Bucket with Versioning nella Guida per l'utente](#). Amazon Simple Storage Service

### Note

Questi frammenti di codice presuppongono che tu abbia compreso il materiale di base e che tu abbia configurato le credenziali predefinite utilizzando le informazioni in AWS [the section called "Configurazione per l'accesso Single Sign-On per l'SDK"](#)

## Argomenti

- [Caricamento di un oggetto](#)
- [Caricamento di oggetti in più parti](#)
- [Eliminazione di un oggetto](#)
- [Elenca oggetti](#)

- [Altri esempi](#)

## Caricamento di un oggetto

Crea un file [PutObjectRequest](#) fornisci un nome di bucket e un nome chiave. Quindi usa il `putObject` metodo di `S3Client` con un [RequestBody](#) che contiene il contenuto dell'oggetto e l'oggetto. `PutObjectRequest` Il bucket deve esistere; in caso contrario il servizio restituirà un errore.

### Importazioni

```
import java.io.IOException;
import java.nio.ByteBuffer;
import java.util.Random;
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.paginators.ListObjectsV2Iterable;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Request;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Response;
import software.amazon.awssdk.services.s3.model.S3Object;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.DeleteObjectRequest;
import software.amazon.awssdk.services.s3.model.DeleteBucketRequest;
import software.amazon.awssdk.services.s3.model.CreateMultipartUploadRequest;
import software.amazon.awssdk.services.s3.model.CreateMultipartUploadResponse;
import software.amazon.awssdk.services.s3.model.CompletedMultipartUpload;
import software.amazon.awssdk.services.s3.model.CreateBucketRequest;
import software.amazon.awssdk.services.s3.model.CompletedPart;
import software.amazon.awssdk.services.s3.model.CreateBucketConfiguration;
import software.amazon.awssdk.services.s3.model.UploadPartRequest;
import software.amazon.awssdk.services.s3.model.CompleteMultipartUploadRequest;
import software.amazon.awssdk.services.s3.waiters.S3Waiter;
import software.amazon.awssdk.services.s3.model.HeadBucketRequest;
import software.amazon.awssdk.services.s3.model.HeadBucketResponse;
```

### Codice

```
Region region = Region.US_WEST_2;
```

```
s3 = S3Client.builder()
    .region(region)
    .build();

createBucket(s3, bucketName, region);

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

s3.putObject(objectRequest,
    RequestBody.fromByteBuffer(getRandomByteBuffer(10_000)));
```

Vedi [l'esempio completo](#) su GitHub

## Caricamento di oggetti in più parti

Usa il `createMultipartUpload` metodo di `S3Client` per ottenere un ID di caricamento. Quindi, utilizzare il metodo `uploadPart` per caricare ogni parte. Infine, usa il `completeMultipartUpload` metodo di `S3Client` per dire Amazon S3 di unire tutte le parti caricate e completare l'operazione di caricamento.

### Importazioni

```
import java.io.IOException;
import java.nio.ByteBuffer;
import java.util.Random;
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.paginators.ListObjectsV2Iterable;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Request;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Response;
import software.amazon.awssdk.services.s3.model.S3Object;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.DeleteObjectRequest;
import software.amazon.awssdk.services.s3.model.DeleteBucketRequest;
import software.amazon.awssdk.services.s3.model.CreateMultipartUploadRequest;
import software.amazon.awssdk.services.s3.model.CreateMultipartUploadResponse;
```



```
import software.amazon.awssdk.services.s3.model.CompletedMultipartUpload;
import software.amazon.awssdk.services.s3.model.CreateBucketRequest;
import software.amazon.awssdk.services.s3.model.CompletedPart;
import software.amazon.awssdk.services.s3.model.CreateBucketConfiguration;
import software.amazon.awssdk.services.s3.model.UploadPartRequest;
import software.amazon.awssdk.services.s3.model.CompleteMultipartUploadRequest;
import software.amazon.awssdk.services.s3.waiters.S3Waiter;
import software.amazon.awssdk.services.s3.model.HeadBucketRequest;
import software.amazon.awssdk.services.s3.model.HeadBucketResponse;
```

## Codice

```
        // First create a multipart upload and get the upload id
        CreateMultipartUploadRequest createMultipartUploadRequest =
CreateMultipartUploadRequest.builder()
            .bucket(bucketName)
            .key(key)
            .build();

        CreateMultipartUploadResponse response =
s3.createMultipartUpload(createMultipartUploadRequest);
        String uploadId = response.uploadId();
        System.out.println(uploadId);

        // Upload all the different parts of the object
        UploadPartRequest uploadPartRequest1 = UploadPartRequest.builder()
            .bucket(bucketName)
            .key(key)
            .uploadId(uploadId)
            .partNumber(1).build();

        String etag1 = s3
            .uploadPart(uploadPartRequest1,
RequestBody.fromByteBuffer(getRandomByteBuffer(5 * mB)))
            .eTag();

        CompletedPart part1 =
CompletedPart.builder().partNumber(1).eTag(etag1).build();

        UploadPartRequest uploadPartRequest2 =
UploadPartRequest.builder().bucket(bucketName).key(key)
            .uploadId(uploadId)
            .partNumber(2).build();
```

```
String etag2 = s3
    .uploadPart(uploadPartRequest2,
RequestBody.fromByteBuffer(getRandomByteBuffer(3 * mB)))
    .eTag();
CompletedPart part2 =
CompletedPart.builder().partNumber(2).eTag(etag2).build();

// Finally call completeMultipartUpload operation to tell S3 to merge
all
// uploaded
// parts and finish the multipart operation.
CompletedMultipartUpload completedMultipartUpload =
CompletedMultipartUpload.builder()
    .parts(part1, part2)
    .build();

CompleteMultipartUploadRequest completeMultipartUploadRequest =
CompleteMultipartUploadRequest.builder()
    .bucket(bucketName)
    .key(key)
    .uploadId(uploadId)
    .multipartUpload(completedMultipartUpload)
    .build();

s3.completeMultipartUpload(completeMultipartUploadRequest);
```

Guarda l'esempio [completo](#) su. GitHub

## Eliminazione di un oggetto

Crea un [DeleteObjectRequest](#) fornisci un nome di bucket e un nome chiave. Usa il `deleteObject` metodo di `S3Client` e passagli il nome di un bucket e di un oggetto da eliminare. Il bucket e la chiave dell'oggetto specificati devono esistere; in caso contrario il servizio restituirà un errore.

## Importazioni

```
import java.io.IOException;
import java.nio.ByteBuffer;
import java.util.Random;
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.paginators.ListObjectsV2Iterable;
```

```
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Request;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Response;
import software.amazon.awssdk.services.s3.model.S3Object;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.DeleteObjectRequest;
import software.amazon.awssdk.services.s3.model.DeleteBucketRequest;
import software.amazon.awssdk.services.s3.model.CreateMultipartUploadRequest;
import software.amazon.awssdk.services.s3.model.CreateMultipartUploadResponse;
import software.amazon.awssdk.services.s3.model.CompletedMultipartUpload;
import software.amazon.awssdk.services.s3.model.CreateBucketRequest;
import software.amazon.awssdk.services.s3.model.CompletedPart;
import software.amazon.awssdk.services.s3.model.CreateBucketConfiguration;
import software.amazon.awssdk.services.s3.model.UploadPartRequest;
import software.amazon.awssdk.services.s3.model.CompleteMultipartUploadRequest;
import software.amazon.awssdk.services.s3.waiters.S3Waiter;
import software.amazon.awssdk.services.s3.model.HeadBucketRequest;
import software.amazon.awssdk.services.s3.model.HeadBucketResponse;
```

## Codice

```
DeleteObjectRequest deleteObjectRequest = DeleteObjectRequest.builder()
    .bucket(bucketName)
    .key(key)
    .build();

s3.deleteObject(deleteObjectRequest);
```

Vedi l'esempio [completo](#) su GitHub

## Copia di un oggetto

Crea [CopyObjectRequest](#) fornisci un nome di bucket in cui è inserito l'oggetto, un valore di stringa con codifica URL (vedi il metodo `URLEncoder.encode`) e il nome chiave dell'oggetto. Usa il metodo di `S3Client` e passa l'oggetto. `copyObject` [CopyObjectRequest](#) Il bucket e la chiave dell'oggetto specificati devono esistere; in caso contrario il servizio restituirà un errore.

## Importazioni

```
import software.amazon.awssdk.regions.Region;
```

```
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.CopyObjectRequest;
import software.amazon.awssdk.services.s3.model.CopyObjectResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;
```

## Codice

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.CopyObjectRequest;
import software.amazon.awssdk.services.s3.model.CopyObjectResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class CopyObject {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <objectKey> <fromBucket> <toBucket>

            Where:
                objectKey - The name of the object (for example, book.pdf).
                fromBucket - The S3 bucket name that contains the object (for
                example, bucket1).
                toBucket - The S3 bucket to copy the object to (for example,
                bucket2).

            """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String objectKey = args[0];
```

```
String fromBucket = args[1];
String toBucket = args[2];
System.out.format("Copying object %s from bucket %s to %s\n", objectKey,
fromBucket, toBucket);
Region region = Region.US_EAST_1;
S3Client s3 = S3Client.builder()
    .region(region)
    .build();

copyBucketObject(s3, fromBucket, objectKey, toBucket);
s3.close();
}

public static String copyBucketObject(S3Client s3, String fromBucket, String
objectKey, String toBucket) {
    CopyObjectRequest copyReq = CopyObjectRequest.builder()
        .sourceBucket(fromBucket)
        .sourceKey(objectKey)
        .destinationBucket(toBucket)
        .destinationKey(objectKey)
        .build();

    try {
        CopyObjectResponse copyRes = s3.copyObject(copyReq);
        return copyRes.copyObjectResult().toString();

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
}
```

Vedi [l'esempio completo](#) su GitHub

## Elenca oggetti

Crea un [ListObjectsRequest](#) fornisci il nome del bucket. Quindi invoca il `listObjects` metodo di `S3Client` e passa l'oggetto. `ListObjectsRequest` Questo metodo restituisce un oggetto [ListObjectsResponse](#) che contiene tutti gli oggetti nel bucket. È possibile richiamare il metodo contenuto di questo oggetto per ottenere un elenco di oggetti. È possibile scorrere questo elenco per visualizzare gli oggetti, come illustrato nell'esempio di codice riportato di seguito.

## Importazioni

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.ListObjectsRequest;
import software.amazon.awssdk.services.s3.model.ListObjectsResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.S3Object;
import java.util.List;
```

## Codice

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.ListObjectsRequest;
import software.amazon.awssdk.services.s3.model.ListObjectsResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.S3Object;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class ListObjects {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <bucketName>\s

            Where:
                bucketName - The Amazon S3 bucket from which objects are read.\s
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }
    }
}
```

```
    }

    String bucketName = args[0];
    Region region = Region.US_EAST_1;
    S3Client s3 = S3Client.builder()
        .region(region)
        .build();

    listBucketObjects(s3, bucketName);
    s3.close();
}

public static void listBucketObjects(S3Client s3, String bucketName) {
    try {
        ListObjectsRequest listObjects = ListObjectsRequest
            .builder()
            .bucket(bucketName)
            .build();

        ListObjectsResponse res = s3.listObjects(listObjects);
        List<S3Object> objects = res.contents();
        for (S3Object myValue : objects) {
            System.out.println("\n The name of the key is " + myValue.key());
            System.out.println("\n The object is " + calKb(myValue.size()) + " KBs");
            System.out.println("\n The owner is " + myValue.owner());
        }

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

// convert bytes to kbs.
private static long calKb(Long val) {
    return val / 1024;
}
}
```

Vedi l'[esempio completo](#) su. GitHub

## Altri esempi

La sezione [Esempi di codice](#) di questa guida contiene altri esempi di utilizzo di oggetti Amazon S3, tra cui come [scaricare un oggetto](#).

## Lavora con URL Amazon S3 prefirmati

Gli URL prefirmati forniscono l'accesso temporaneo agli oggetti S3 privati senza richiedere agli utenti di disporre di AWS credenziali o autorizzazioni.

Ad esempio, si supponga che Alice abbia accesso a un oggetto S3 e che voglia condividere temporaneamente l'accesso a quell'oggetto con Bob. Alice può generare una richiesta GET prefirmata da condividere con Bob in modo che possa scaricare l'oggetto senza richiedere l'accesso alle credenziali di Alice. È possibile generare URL prefirmati per HTTP GET e per richieste HTTP PUT.

### Genera un URL prefirmato per un oggetto, quindi scaricalo (richiesta GET)

L'esempio seguente è composto da due parti.

- Parte 1: Alice genera l'URL prefirmato per un oggetto.
- Parte 2: Bob scarica l'oggetto utilizzando l'URL prefirmato.

#### Parte 1: Generazione dell'URL

Alice ha già un oggetto in un bucket S3. Usa il codice seguente per generare una stringa URL che Bob può utilizzare in una successiva richiesta GET.

#### Importazioni

```
import com.example.s3.util.PresignUrlUtils;
import org.slf4j.Logger;
import software.amazon.awssdk.http.HttpExecuteRequest;
import software.amazon.awssdk.http.HttpExecuteResponse;
import software.amazon.awssdk.http.SdkHttpClient;
import software.amazon.awssdk.http.SdkHttpMethod;
import software.amazon.awssdk.http.SdkHttpRequest;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
```



```
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.presigner.S3Presigner;
import software.amazon.awssdk.services.s3.presigner.model.GetObjectPresignRequest;
import software.amazon.awssdk.services.s3.presigner.model.PresignedGetObjectRequest;
import software.amazon.awssdk.utils.IoUtils;

import java.io.ByteArrayOutputStream;
import java.io.File;
import java.io.IOException;
import java.io.InputStream;
import java.net.HttpURLConnection;
import java.net.URISyntaxException;
import java.net.URL;
import java.net.http.HttpClient;
import java.net.http.HttpRequest;
import java.net.http.HttpResponse;
import java.nio.file.Paths;
import java.time.Duration;
import java.util.UUID;

/* Create a pre-signed URL to download an object in a subsequent GET request. */
public String createPresignedGetUrl(String bucketName, String keyName) {
    try (S3Presigner presigner = S3Presigner.create()) {

        GetObjectRequest objectRequest = GetObjectRequest.builder()
            .bucket(bucketName)
            .key(keyName)
            .build();

        GetObjectPresignRequest presignRequest = GetObjectPresignRequest.builder()
            .signatureDuration(Duration.ofMinutes(10)) // The URL will expire
in 10 minutes.
            .getObjectRequest(objectRequest)
            .build();

        PresignedGetObjectRequest presignedRequest =
presigner.presignGetObject(presignRequest);
        logger.info("Presigned URL: [{}]", presignedRequest.url().toString());
        logger.info("HTTP method: [{}]", presignedRequest.httpRequest().method());

        return presignedRequest.url().toExternalForm();
    }
}
```

## Parte 2: scarica l'oggetto

Bob utilizza una delle seguenti tre opzioni di codice per scaricare l'oggetto. In alternativa, potrebbe utilizzare un browser per eseguire la richiesta GET.

### Usa JDK `HttpURLConnection` (dalla v1.1)

```
/* Use the JDK HttpURLConnection (since v1.1) class to do the download. */
public byte[] useHttpURLConnectionToGet(String presignedUrlString) {
    ByteArrayOutputStream byteArrayOutputStream = new ByteArrayOutputStream(); //
    Capture the response body to a byte array.

    try {
        URL presignedUrl = new URL(presignedUrlString);
        HttpURLConnection connection = (HttpURLConnection)
presignedUrl.openConnection();
        connection.setRequestMethod("GET");
        // Download the result of executing the request.
        try (InputStream content = connection.getInputStream()) {
            IoUtils.copy(content, byteArrayOutputStream);
        }
        logger.info("HTTP response code is " + connection.getResponseCode());

    } catch (S3Exception | IOException e) {
        logger.error(e.getMessage(), e);
    }
    return byteArrayOutputStream.toByteArray();
}
```

### Usa `HttpClient` JDK (dalla v11)

```
/* Use the JDK HttpClient (since v11) class to do the download. */
public byte[] useHttpClientToGet(String presignedUrlString) {
    ByteArrayOutputStream byteArrayOutputStream = new ByteArrayOutputStream(); //
    Capture the response body to a byte array.

    HttpRequest.Builder requestBuilder = HttpRequest.newBuilder();
    HttpClient httpClient = HttpClient.newHttpClient();
    try {
        URL presignedUrl = new URL(presignedUrlString);
        HttpResponse<InputStream> response = httpClient.send(requestBuilder
        .uri(presignedUrl.toURI())
        .GET())
    }
```

```

        .build(),
        HttpResponse.BodyHandlers.ofInputStream());

    IoUtils.copy(response.body(), byteArrayOutputStream);

    logger.info("HTTP response code is " + response.statusCode());
} catch (URISyntaxException | InterruptedException | IOException e) {
    logger.error(e.getMessage(), e);
}
return byteArrayOutputStream.toByteArray();
}

```

## Utilizzo `SdkHttpClient` dall'SDK for Java

```

/* Use the AWS SDK for Java SdkHttpClient class to do the download. */
public byte[] useSdkHttpClientToPut(String presignedUrlString) {

    ByteArrayOutputStream byteArrayOutputStream = new ByteArrayOutputStream(); //
    Capture the response body to a byte array.
    try {
        URL presignedUrl = new URL(presignedUrlString);
        SdkHttpRequest request = SdkHttpRequest.builder()
            .method(SdkHttpMethod.GET)
            .uri(presignedUrl.toURI())
            .build();

        HttpExecuteRequest executeRequest = HttpExecuteRequest.builder()
            .request(request)
            .build();

        try (SdkHttpClient sdkHttpClient = ApacheHttpClient.create()) {
            HttpExecuteResponse response =
            sdkHttpClient.prepareRequest(executeRequest).call();
            response.responseBody().ifPresentOrElse(
                abortableInputStream -> {
                    try {
                        IoUtils.copy(abortableInputStream,
byteArrayOutputStream);
                    } catch (IOException e) {
                        throw new RuntimeException(e);
                    }
                },
            },

```

```
        () -> logger.error("No response body.");

        logger.info("HTTP Response code is {}",
response.httpResponse().statusCode());
    }
} catch (URISyntaxException | IOException e) {
    logger.error(e.getMessage(), e);
}
return byteArrayOutputStream.toByteArray();
}
```

Guarda l'[esempio completo e prosegui il test](#). [GitHub](#)

## Genera un URL prefirmato per un caricamento, quindi carica un file (richiesta PUT)

L'esempio seguente è composto da due parti.

- Parte 1: Alice genera l'URL prefirmato per caricare un oggetto.
- Parte 2: Bob carica un file utilizzando l'URL prefirmato.

### Parte 1: Generazione dell'URL

Alice ha già un bucket S3. Usa il codice seguente per generare una stringa URL che Bob può utilizzare in una successiva richiesta PUT.

### Importazioni

```
import com.example.s3.util.PresignUrlUtils;
import org.slf4j.Logger;
import software.amazon.awssdk.core.internal.sync.FileContentStreamProvider;
import software.amazon.awssdk.http.HttpExecuteRequest;
import software.amazon.awssdk.http.HttpExecuteResponse;
import software.amazon.awssdk.http.SdkHttpClient;
import software.amazon.awssdk.http.SdkHttpMethod;
import software.amazon.awssdk.http.SdkHttpRequest;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.presigner.S3Presigner;
import software.amazon.awssdk.services.s3.presigner.model.PresignedPutObjectRequest;
import software.amazon.awssdk.services.s3.presigner.model.PutObjectPresignRequest;
```

```
import java.io.File;
import java.io.IOException;
import java.io.OutputStream;
import java.io.RandomAccessFile;
import java.net.HttpURLConnection;
import java.net.URISyntaxException;
import java.net.URL;
import java.net.http.HttpClient;
import java.net.http.HttpRequest;
import java.net.http.HttpResponse;
import java.nio.ByteBuffer;
import java.nio.channels.FileChannel;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.time.Duration;
import java.util.Map;
import java.util.UUID;
```

```
/* Create a presigned URL to use in a subsequent PUT request */
public String createPresignedUrl(String bucketName, String keyName, Map<String,
String> metadata) {
    try (S3Presigner presigner = S3Presigner.create()) {

        PutObjectRequest objectRequest = PutObjectRequest.builder()
            .bucket(bucketName)
            .key(keyName)
            .metadata(metadata)
            .build();

        PutObjectPresignRequest presignRequest = PutObjectPresignRequest.builder()
            .signatureDuration(Duration.ofMinutes(10)) // The URL expires in
10 minutes.

            .putObjectRequest(objectRequest)
            .build();

        PresignedPutObjectRequest presignedRequest =
presigner.presignPutObject(presignRequest);
        String myURL = presignedRequest.url().toString();
        logger.info("Presigned URL to upload a file to: [{}]", myURL);
        logger.info("HTTP method: [{}]", presignedRequest.httpRequest().method());
```

```

        return presignedRequest.url().toExternalForm();
    }
}

```

## Parte 2: carica un oggetto file

Bob utilizza una delle seguenti tre opzioni di codice per caricare un file.

### Usa JDK **URLConnection** (dalla v1.1)

```

/* Use the JDK HttpURLConnection (since v1.1) class to do the upload. */
public void useURLConnectionToPut(String presignedUrlString, File fileToPut,
Map<String, String> metadata) {
    logger.info("Begin [{}] upload", fileToPut.toString());
    try {
        URL presignedUrl = new URL(presignedUrlString);
        HttpURLConnection connection = (HttpURLConnection)
presignedUrl.openConnection();
        connection.setDoOutput(true);
        metadata.forEach((k, v) -> connection.setRequestProperty("x-amz-meta-" + k,
v));

        connection.setRequestMethod("PUT");
        OutputStream out = connection.getOutputStream();

        try (RandomAccessFile file = new RandomAccessFile(fileToPut, "r");
            FileChannel inChannel = file.getChannel()) {
            ByteBuffer buffer = ByteBuffer.allocate(8192); //Buffer size is 8k

            while (inChannel.read(buffer) > 0) {
                buffer.flip();
                for (int i = 0; i < buffer.limit(); i++) {
                    out.write(buffer.get());
                }
                buffer.clear();
            }
        } catch (IOException e) {
            logger.error(e.getMessage(), e);
        }

        out.close();
        connection.getResponseCode();
        logger.info("HTTP response code is " + connection.getResponseCode());

    } catch (S3Exception | IOException e) {

```

```

        logger.error(e.getMessage(), e);
    }
}

```

## Usa **HttpClient** JDK (dalla v11)

```

/* Use the JDK HttpClient (since v11) class to do the upload. */
public void useHttpClientToPut(String presignedUrlString, File fileToPut,
Map<String, String> metadata) {
    logger.info("Begin [{}] upload", fileToPut.toString());

    HttpRequest.Builder requestBuilder = HttpRequest.newBuilder();
    metadata.forEach((k, v) -> requestBuilder.header("x-amz-meta-" + k, v));

    HttpClient httpClient = HttpClient.newHttpClient();
    try {
        final HttpResponse<Void> response = httpClient.send(requestBuilder
            .uri(new URL(presignedUrlString).toURI())
            .PUT(HttpRequest.BodyPublishers.ofFile(Path.of(fileToPut.toURI())))
            .build(),
            HttpResponse.BodyHandlers.discarding());

        logger.info("HTTP response code is " + response.statusCode());

    } catch (URISyntaxException | InterruptedException | IOException e) {
        logger.error(e.getMessage(), e);
    }
}

```

## Utilizzo **SdkHttpClient** dall'SDK for Java

```

/* Use the AWS SDK for Java V2 SdkHttpClient class to do the upload. */
public void useSdkHttpClientToPut(String presignedUrlString, File fileToPut,
Map<String, String> metadata) {
    logger.info("Begin [{}] upload", fileToPut.toString());

    try {
        URL presignedUrl = new URL(presignedUrlString);

        SdkHttpRequest.Builder requestBuilder = SdkHttpRequest.builder()
            .method(SdkHttpMethod.PUT)
            .uri(presignedUrl.toURI());
    }
}

```

```
// Add headers
metadata.forEach((k, v) -> requestBuilder.putHeader("x-amz-meta-" + k, v));
// Finish building the request.
SdkHttpRequest request = requestBuilder.build();

HttpExecuteRequest executeRequest = HttpExecuteRequest.builder()
    .request(request)
    .contentStreamProvider(new
FileContentStreamProvider(fileToPut.toPath()))
    .build();

try (SdkHttpClient sdkHttpClient = ApacheHttpClient.create()) {
    HttpExecuteResponse response =
sdkHttpClient.prepareRequest(executeRequest).call();
    logger.info("Response code: {}", response.httpResponse().statusCode());
}
} catch (URISyntaxException | IOException e) {
    logger.error(e.getMessage(), e);
}
}
```

Guarda l'[esempio completo e prosegui il test](#). GitHub

## Accesso interregionale per Amazon S3

Quando lavori con i bucket Amazon Simple Storage Service (Amazon S3) Simple Storage Service (Amazon S3), di solito sai qual Regione AWS è il bucket giusto. La regione con cui lavori viene determinata al momento della creazione del client S3.

Tuttavia, a volte potresti dover lavorare con un bucket specifico, ma non sai se si trova nella stessa regione impostata per il client S3.

Invece di effettuare più chiamate per determinare la regione del bucket, puoi utilizzare l'SDK per abilitare l'accesso ai bucket S3 in diverse regioni.

### Installazione

Il supporto per l'accesso interregionale è diventato disponibile con la versione 2.20.111 dell'SDK. Usa questa versione o una successiva nel tuo file di build di Maven per la s3 dipendenza, come mostrato nel seguente frammento.

```
<dependency>
```



```
<groupId>software.amazon.awssdk</groupId>
<artifactId>s3</artifactId>
<version>2.20.111</version>
</dependency>
```

Successivamente, quando crei il tuo client S3, abilita l'accesso interregionale come mostrato nello snippet. Per impostazione predefinita, l'accesso non è abilitato.

```
S3AsyncClient client = S3AsyncClient.builder()
    .crossRegionAccessEnabled(true)
    .build();
```

## In che modo l'SDK fornisce l'accesso a più regioni

Quando si fa riferimento a un bucket esistente in una richiesta, ad esempio quando si utilizza il `putObject` metodo, l'SDK avvia una richiesta alla regione configurata per il client.

Se il bucket non esiste in quella regione specifica, la risposta all'errore include la regione effettiva in cui risiede il bucket. L'SDK utilizza quindi la regione corretta in una seconda richiesta.

Per ottimizzare le richieste future allo stesso bucket, l'SDK memorizza nella cache questa mappatura della regione nel client.

## Considerazioni

Quando abiliti l'accesso ai bucket tra regioni, tieni presente che la prima chiamata API potrebbe comportare un aumento della latenza se il bucket non si trova nella regione configurata dal client. Tuttavia, le chiamate successive traggono vantaggio dalle informazioni sulla regione memorizzate nella cache, con conseguente miglioramento delle prestazioni.

Quando si abilita l'accesso tra regioni, l'accesso al bucket non viene influenzato. L'utente deve essere autorizzato ad accedere al bucket in qualsiasi regione in cui risieda.

Amazon Simple Storage Service (Amazon S3) Simple Storage Service (Amazon S3) offre la possibilità di specificare un checksum quando carichi un oggetto. Quando specifichi un checksum, questo viene memorizzato con l'oggetto e può essere convalidato quando l'oggetto viene scaricato.

I checksum forniscono un ulteriore livello di integrità dei dati durante il trasferimento dei file. Con i checksum, è possibile verificare la coerenza dei dati confermando che il file ricevuto corrisponde al

file originale. Per ulteriori informazioni sui checksum con Amazon S3, consulta la Amazon [Simple Storage Service User Guide](#).

Amazon S3 attualmente supporta quattro algoritmi di checksum: SHA-1, SHA-256, CRC-32 e CRC-32C. Hai la flessibilità di scegliere l'algoritmo più adatto alle tue esigenze e lasciare che sia l'SDK a calcolare il checksum. In alternativa, puoi specificare il proprio valore di checksum precalcolato utilizzando uno dei quattro algoritmi supportati.

Discutiamo i checksum in due fasi di richiesta: caricamento di un oggetto e download di un oggetto.

## Caricamento di un oggetto

I valori validi per l'algoritmo sono CRC32, CRC32C, SHA1 e SHA256

Il seguente frammento di codice mostra una richiesta di caricamento di un oggetto con un checksum CRC-32. Quando l'SDK invia la richiesta, calcola il checksum CRC-32 e carica l'oggetto. Amazon S3 memorizza il checksum con l'oggetto.

Se il checksum calcolato dall'SDK non corrisponde al checksum calcolato da Amazon S3 quando riceve la richiesta, viene restituito un errore.

Utilizza un valore di checksum precalcolato

Un valore di checksum precalcolato fornito con la richiesta disabilita il calcolo automatico da parte dell'SDK e utilizza invece il valore fornito.

L'esempio seguente mostra una richiesta con un checksum SHA-256 precalcolato.

Se Amazon S3 determina che il valore del checksum non è corretto per l'algoritmo specificato, il servizio restituisce una risposta di errore.

Caricamenti in più parti

Puoi anche utilizzare i checksum con caricamenti in più parti.

## Download di un oggetto

Quando utilizzate il metodo [getObject](#) per scaricare un oggetto, l'SDK convalida automaticamente il checksum . `enabled`

La richiesta nel seguente frammento indica all'SDK di convalidare il checksum nella risposta calcolando il checksum e confrontando i valori.

Se l'oggetto non è stato caricato con un checksum, non viene effettuata alcuna convalida.

Un oggetto in Amazon S3 può avere più checksum, ma al download viene convalidato solo un checksum. La seguente precedenza, basata sull'efficienza dell'algoritmo di checksum, determina quale checksum viene convalidato dall'SDK:

1. CRC-32C
2. CRC-32
3. SHA-1
4. SHA-256

Ad esempio, se una risposta contiene entrambi i checksum CRC-32 e SHA-256, viene convalidato solo il checksum CRC-32.

## Usa un client S3 performante: client S3 basato su AWS CRT

Il client S3 AWS basato su CRT, basato su [AWSCommon Runtime \(CRT\)](#), è un client asincrono S3 alternativo. [Trasferisce oggetti da e verso Amazon Simple Storage Service \(Amazon S3\) con prestazioni e affidabilità migliorate utilizzando automaticamente l'API di caricamento multipart e i recuperi per intervallo di byte di Amazon S3.](#)

Il client S3 AWS basato su CRT migliora l'affidabilità del trasferimento in caso di guasto della rete. L'affidabilità è migliorata riprovando le singole parti non riuscite di un trasferimento di file senza riavviare il trasferimento dall'inizio.

Inoltre, il client S3 AWS basato su CRT offre un pool di connessioni avanzato e un bilanciamento del carico DNS (Domain Name System), che migliora anche la velocità effettiva.

Puoi utilizzare il client S3 AWS basato su CRT al posto del client asincrono S3 standard dell'SDK e sfruttare subito il suo throughput migliorato.

### AWSComponenti basati su CRT nell'SDK

Il client S3 AWS basato su CRT, descritto in questo argomento, e il client HTTP AWS basato su CRT sono componenti diversi dell'SDK.

Il client S3 AWS basato su CRT è un'implementazione dell'AsyncClientinterfaccia [S3](#) e viene utilizzato per lavorare con il servizio Amazon S3. È un'alternativa all'implementazione dell'interfaccia basata su Java e offre diversi vantaggi. `S3AsyncClient`

Il [client HTTP AWS basato su CRT](#) è un'implementazione dell'[SdkAsyncHttpClient](#) interfaccia e viene utilizzato per la comunicazione HTTP generale. È un'alternativa all'implementazione Netty dell'[SdkAsyncHttpClient](#) interfaccia e offre diversi vantaggi.

Sebbene entrambi i componenti utilizzino librerie di [AWSCommon Runtime](#), il client S3 AWS basato su CRT utilizza la [libreria aws-c-s 3 e supporta le funzionalità](#) dell'API di caricamento multipart [S3](#). Poiché il client HTTP AWS basato su CRT è pensato per un uso generico, non supporta le funzionalità dell'API di caricamento multipart S3.

## Aggiungi dipendenze per utilizzare il client S3 basato su CRT AWS

Per utilizzare il client S3 AWS basato su CRT, aggiungi le seguenti due dipendenze al tuo file di progetto Maven. L'esempio mostra le versioni minime da utilizzare. [Cerca nel repository centrale di Maven le versioni più recenti degli artefatti s3 e aws-crt.](#)

```
<dependency>
  <groupId>software.amazon.awssdk</groupId>
  <artifactId>s3</artifactId>
  <version>2.20.68</version>
</dependency>
<dependency>
  <groupId>software.amazon.awssdk.crt</groupId>
  <artifactId>aws-crt</artifactId>
  <version>0.21.16</version>
</dependency>
```

## Crea un'istanza AWS del client S3 basato su CRT

Crea un'istanza del client S3 AWS basato su CRT con le impostazioni predefinite, come mostrato nel seguente frammento di codice.

```
S3AsyncClient s3AsyncClient = S3AsyncClient.crtCreate();
```

Per configurare il client, utilizzate il generatore di client CRT. AWS È possibile passare dal client asincrono S3 standard al client basato su AWS CRT modificando il metodo del generatore.

```
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3AsyncClient;
```

```
S3AsyncClient s3AsyncClient =
    S3AsyncClient.crtBuilder()
        .credentialsProvider(DefaultCredentialsProvider.create())
        .region(Region.US_WEST_2)
        .targetThroughputInGbps(20.0)
        .minimumPartSizeInBytes(8 * 1025 * 1024L)
        .build();
```

### Note

Alcune impostazioni del generatore standard potrebbero non essere attualmente supportate nel generatore del client CRT. AWS Scarica lo standard builder chiamando `S3AsyncClient#builder()`

## Usa il client S3 AWS basato su CRT

Usa il client S3 AWS basato su CRT per chiamare le operazioni dell'API Amazon S3. L'esempio seguente illustra le operazioni [PutObject](#) e [GetObject](#) disponibili tramite AWS SDK for Java

```
import software.amazon.awssdk.core.async.AsyncRequestBody;
import software.amazon.awssdk.core.async.AsyncResponseTransformer;
import software.amazon.awssdk.services.s3.S3AsyncClient;
import software.amazon.awssdk.services.s3.model.GetObjectResponse;
import software.amazon.awssdk.services.s3.model.PutObjectResponse;

S3AsyncClient s3Client = S3AsyncClient.crtCreate();

// Upload a local file to Amazon S3.
PutObjectResponse putObjectResponse =
    s3Client.putObject(req -> req.bucket(<BUCKET_NAME>)
        .key(<KEY_NAME>),
        AsyncRequestBody.fromFile(Paths.get(<FILE_NAME>)))
        .join();

// Download an object from Amazon S3 to a local file.
GetObjectResponse getObjectResponse =
    s3Client.getObject(req -> req.bucket(<BUCKET_NAME>)
        .key(<KEY_NAME>),
```

```
AsyncResponseTransformer.toFile(Paths.get(<FILE_NAME>)))  
.join();
```

## Trasferisci file e directory con Amazon S3 Transfer Manager

Amazon S3 Transfer Manager è un'utilità di trasferimento file open source di alto livello per. AWS SDK for Java 2.x Usalo per trasferire file e directory da e verso Amazon Simple Storage Service (Amazon S3).

[Se basato sul client S3 AWS basato su CRT, S3 Transfer Manager può sfruttare i miglioramenti delle prestazioni, come l'API di caricamento multiparte e i recuperi a intervalli di byte.](#)

Con S3 Transfer Manager, puoi anche monitorare l'avanzamento di un trasferimento in tempo reale e mettere in pausa il trasferimento per un'esecuzione successiva.

### Inizia a usare

Aggiungi dipendenze al tuo file di build

Per utilizzare S3 Transfer Manager con prestazioni migliorate basate sul client S3 basato su AWS CRT, configura il file di build con le seguenti dipendenze.

- Utilizzate la versione **2.19.1** o successiva dell'SDK for Java 2.x.
- Aggiungi l'`s3-transfer-manager` artefatto come dipendenza.
- *Aggiungi l'`aws-crt` artefatto come dipendenza nella versione **0.20.3** o successiva.*

Il seguente esempio di codice mostra come configurare le dipendenze del progetto per Maven.

```
<project>  
  <properties>  
    <aws.sdk.version>2.19.1</aws.sdk.version>  
  </properties>  
  <dependencyManagement>  
    <dependencies>  
      <dependency>  
        <groupId>software.amazon.awssdk</groupId>  
        <artifactId>bom</artifactId>
```

```
<version>${aws.sdk.version}</version>
<type>pom</type>
<scope>import</scope>
</dependency>
</dependencies>
</dependencyManagement>
<dependencies>
<dependency>
  <groupId>software.amazon.awssdk</groupId>
  <artifactId>s3-transfer-manager</artifactId>
</dependency>
<dependency>
  <groupId>software.amazon.awssdk.crt</groupId>
  <artifactId>aws-crt</artifactId>
  <version>0.20.3</version>
</dependency>
</dependencies>
</project>
```

[Cerca nel repository centrale di Maven le versioni più recenti degli artefatti s3-transfer-manager e aws-crt.](#)

## Crea un'istanza di S3 Transfer Manager

Il seguente frammento mostra come creare un'istanza di [S3](#) con impostazioni predefinite.

```
S3TransferManager transferManager = S3TransferManager.create();
```

L'esempio seguente mostra come configurare un S3 Transfer Manager con impostazioni personalizzate. In questo esempio, un'istanza di [S3 AWS basata su CRT](#) viene utilizzata come client sottostante per S3 Transfer Manager.

```
S3AsyncClient s3AsyncClient = S3AsyncClient.crtBuilder()
    .credentialsProvider(DefaultCredentialsProvider.create())
    .region(Region.US_EAST_1)
    .targetThroughputInGbps(20.0)
    .minimumPartSizeInBytes(8 * MB)
    .build();

S3TransferManager transferManager = S3TransferManager.builder()
    .s3Client(s3AsyncClient)
```

```
.build();
```

### Note

Se la `aws-crt` dipendenza non è inclusa nel file di build, S3 Transfer Manager è basato sul client asincrono S3 standard utilizzato nell'SDK for Java 2.x.

## Carica un file in un bucket S3

L'esempio seguente mostra un esempio di caricamento di file insieme all'uso opzionale di [LoggingTransferListener](#), che registra l'avanzamento del caricamento.

[Per caricare un file su Amazon S3 utilizzando S3 Transfer Manager, passa un UploadFileRequest oggetto al S3TransferManager metodo uploadFile.](#)

L'[FileUpload](#) oggetto restituito dal `uploadFile` metodo rappresenta il processo di caricamento. Al termine della richiesta, l'[CompletedFileUpload](#) oggetto contiene informazioni sul caricamento.

```
public String uploadFile(S3TransferManager transferManager, String bucketName,
                        String key, URI filePathURI) {
    UploadFileRequest uploadFileRequest = UploadFileRequest.builder()
        .putObjectRequest(b -> b.bucket(bucketName).key(key))
        .source(Paths.get(filePathURI))
        .build();

    FileUpload fileUpload = transferManager.uploadFile(uploadFileRequest);

    CompletedFileUpload uploadResult = fileUpload.completionFuture().join();
    return uploadResult.response().eTag();
}
```

## Importazioni

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.CompletedFileUpload;
import software.amazon.awssdk.transfer.s3.model.FileUpload;
import software.amazon.awssdk.transfer.s3.model.UploadFileRequest;
```



```
import software.amazon.awssdk.transfer.s3.progress.LoggingTransferListener;
import java.net.URI;
import java.net.URISyntaxException;
import java.net.URL;
import java.nio.file.Paths;
import java.util.UUID;
```

## Scarica un file da un bucket S3

L'esempio seguente mostra un esempio di download insieme all'uso opzionale di [LoggingTransferListener](#), che registra l'avanzamento del download.

[Per scaricare un oggetto da un bucket S3 utilizzando S3 Transfer Manager, crea un DownloadFileRequest oggetto e passalo al metodo DownloadFile.](#)

L'[FileDownload](#) oggetto restituito dal `downloadFile` metodo `S3TransferManager`'s rappresenta il trasferimento del file. Una volta completato il download, [CompletedFileDownload](#) contiene l'accesso alle informazioni sul download.

```
public Long downloadFile(S3TransferManager transferManager, String bucketName,
                        String key, String downloadedFilePath) {
    DownloadFileRequest downloadFileRequest = DownloadFileRequest.builder()
        .getObjectRequest(b -> b.bucket(bucketName).key(key))
        .destination(Paths.get(downloadedFilePath))
        .build();

    FileDownload downloadFile = transferManager.downloadFile(downloadFileRequest);

    CompletedFileDownload downloadResult = downloadFile.completionFuture().join();
    logger.info("Content length [{}]", downloadResult.response().contentLength());
    return downloadResult.response().contentLength();
}
```

## Importazioni

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.CompletedFileDownload;
import software.amazon.awssdk.transfer.s3.model.DownloadFileRequest;
```

```
import software.amazon.awssdk.transfer.s3.model.FileDownload;
import software.amazon.awssdk.transfer.s3.progress.LoggingTransferListener;

import java.io.IOException;
import java.net.URISyntaxException;
import java.net.URL;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.UUID;
```

## Copiare un oggetto Amazon S3 in un altro bucket

L'esempio seguente mostra come copiare un oggetto con S3 Transfer Manager.

Per iniziare la copia di un oggetto da un bucket S3 a un altro bucket, crea un'istanza di base.

### [CopyObjectRequest](#)

Successivamente, racchiudi il file di base `CopyObjectRequest` in un file [CopyRequest](#) che possa essere utilizzato da S3 Transfer Manager.

L'oggetto restituito dal `copy` metodo `S3TransferManager`'s rappresenta il processo di copia. Una volta completato il processo di copia, l'[CompletedCopy](#) oggetto contiene i dettagli sulla risposta.

```
public String copyObject(S3TransferManager transferManager, String bucketName,
    String key, String destinationBucket, String destinationKey) {
    CopyObjectRequest copyObjectRequest = CopyObjectRequest.builder()
        .sourceBucket(bucketName)
        .sourceKey(key)
        .destinationBucket(destinationBucket)
        .destinationKey(destinationKey)
        .build();

    CopyRequest copyRequest = CopyRequest.builder()
        .copyObjectRequest(copyObjectRequest)
        .build();

    Copy copy = transferManager.copy(copyRequest);

    CompletedCopy completedCopy = copy.completionFuture().join();
    return completedCopy.response().copyObjectResult().eTag();
}
```

**Note**

Per eseguire una copia su più regioni con S3 Transfer Manager, abilitalo sul generatore di client S3 AWS basato `crossRegionAccessEnabled` su CRT, come mostrato nel frammento seguente.

```
S3AsyncClient s3AsyncClient = S3AsyncClient.crtBuilder()
    .crossRegionAccessEnabled(true)
    .build();

S3TransferManager transferManager = S3TransferManager.builder()
    .s3Client(s3AsyncClient)
    .build();
```

**Importazioni**

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.services.s3.model.CopyObjectRequest;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.CompletedCopy;
import software.amazon.awssdk.transfer.s3.model.Copy;
import software.amazon.awssdk.transfer.s3.model.CopyRequest;

import java.util.UUID;
```

**Carica una directory locale in un bucket S3**

L'esempio seguente mostra come caricare una directory locale su S3.

Inizia chiamando il metodo [uploadDirectory](#) dell'`S3TransferManager` istanza, passando un [UploadDirectoryRequest](#)

L'[DirectoryUpload](#) oggetto rappresenta il processo di caricamento, che genera un [CompletedDirectoryUpload](#) termine della richiesta. L'`CompletedDirectoryUpload` oggetto contiene informazioni sui risultati del trasferimento, inclusi i file non riusciti a trasferire.

```
public Integer uploadDirectory(S3TransferManager transferManager,
    URI sourceDirectory, String bucketName) {
```

```
DirectoryUpload directoryUpload =
transferManager.uploadDirectory(UploadDirectoryRequest.builder()
    .source(Paths.get(sourceDirectory))
    .bucket(bucketName)
    .build());

CompletedDirectoryUpload completedDirectoryUpload =
directoryUpload.completionFuture().join();
completedDirectoryUpload.failedTransfers()
    .forEach(fail -> logger.warn("Object [{}] failed to transfer",
fail.toString()));
return completedDirectoryUpload.failedTransfers().size();
}
```

## Importazioni

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.s3.model.ObjectIdentifier;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.CompletedDirectoryUpload;
import software.amazon.awssdk.transfer.s3.model.DirectoryUpload;
import software.amazon.awssdk.transfer.s3.model.UploadDirectoryRequest;

import java.net.URI;
import java.net.URISyntaxException;
import java.net.URL;
import java.nio.file.Paths;
import java.util.UUID;
```

## Scarica gli oggetti del bucket S3 in una directory locale

Puoi scaricare gli oggetti in un bucket S3 in una directory locale come mostrato nell'esempio seguente.

Per scaricare gli oggetti in un bucket S3 in una directory locale, iniziate chiamando il metodo [DownloadDirectory](#) di Transfer Manager, passando un [DownloadDirectoryRequest](#)

L'[DirectoryDownload](#) oggetto rappresenta il processo di download, che genera un messaggio [CompletedDirectoryDownload](#) quando la richiesta viene completata.

L'[CompleteDirectoryDownload](#) oggetto contiene informazioni sui risultati del trasferimento, inclusi i file non riusciti a trasferire.

```
public Integer downloadObjectsToDirectory(S3TransferManager transferManager,
    URI destinationPathURI, String bucketName) {
    DirectoryDownload directoryDownload =
transferManager.downloadDirectory(DownloadDirectoryRequest.builder()
    .destination(Paths.get(destinationPathURI))
    .bucket(bucketName)
    .build());
    CompletedDirectoryDownload completedDirectoryDownload =
directoryDownload.completionFuture().join();

    completedDirectoryDownload.failedTransfers()
        .forEach(fail -> logger.warn("Object [{}] failed to transfer",
fail.toString()));
    return completedDirectoryDownload.failedTransfers().size();
}
```

## Importazioni

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.services.s3.model.ObjectIdentifier;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.CompletedDirectoryDownload;
import software.amazon.awssdk.transfer.s3.model.DirectoryDownload;
import software.amazon.awssdk.transfer.s3.model.DownloadDirectoryRequest;
import java.io.IOException;
import java.net.URI;
import java.net.URISyntaxException;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.HashSet;
import java.util.Set;
import java.util.UUID;
import java.util.stream.Collectors;
```

## Vedi esempi completi

[GitHub contiene il codice completo](#) per tutti gli esempi in questa pagina.

# Utilizzo delle Amazon Simple Notification Service

Con Amazon Simple Notification Service, puoi inviare facilmente messaggi di notifica in tempo reale dalle applicazioni agli abbonati su più canali di comunicazione. In questo argomento viene descritto come eseguire alcune delle funzioni di base di Amazon SNS.

## Creazione di un argomento

Un argomento è un raggruppamento logico di canali di comunicazione che definisce i sistemi a cui inviare un messaggio, ad esempio, il fanning out di un messaggio AWS Lambda e un webhook HTTP. Prima invii i messaggi a Amazon SNS, poi tali messaggi vengono distribuiti ai canali definiti nell'argomento. Questo rende i messaggi disponibili per gli abbonati.

Per creare un argomento, crea prima un [CreateTopicRequest](#) oggetto, con il nome dell'argomento impostato utilizzando il `name()` metodo del generatore. Quindi, invia l'oggetto richiesto a Amazon SNS utilizzando il metodo `createTopic()` di [SnsClient](#). È possibile acquisire il risultato di questa richiesta come [CreateTopicResponse](#) oggetto, come dimostrato nel seguente frammento di codice.

### Importazioni

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.CreateTopicRequest;
import software.amazon.awssdk.services.sns.model.CreateTopicResponse;
import software.amazon.awssdk.services.sns.model.SnsException;
```

### Codice

```
public static String createSNSTopic(SnsClient snsClient, String topicName ) {

    CreateTopicResponse result = null;
    try {
        CreateTopicRequest request = CreateTopicRequest.builder()
            .name(topicName)
            .build();

        result = snsClient.createTopic(request);
        return result.topicArn();
    } catch (SnsException e) {

        System.err.println(e.awsErrorDetails().errorMessage());
    }
}
```

```
        System.exit(1);
    }
    return "";
}
```

Vedi l'[esempio completo](#) su GitHub

## Elenca i tuoi Amazon SNS argomenti

Per recuperare un elenco dei tuoi Amazon SNS argomenti esistenti, crea un [ListTopicsRequest](#) oggetto. Quindi, invia l'oggetto richiesto a Amazon SNS utilizzando il metodo `listTopics()` di `SnsClient`. Puoi acquisire il risultato di questa richiesta come [ListTopicsResponse](#) oggetto.

Il frammento di codice seguente stampa il codice di stato HTTP della richiesta e un elenco di Amazon Resource Name (ARN) per i tuoi argomenti Amazon SNS.

### Importazioni

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.ListTopicsRequest;
import software.amazon.awssdk.services.sns.model.ListTopicsResponse;
import software.amazon.awssdk.services.sns.model.SnsException;
```

### Codice

```
public static void listSNSTopics(SnsClient snsClient) {

    try {
        ListTopicsRequest request = ListTopicsRequest.builder()
            .build();

        ListTopicsResponse result = snsClient.listTopics(request);
        System.out.println("Status was " + result.sdkHttpResponse().statusCode() +
            "\n\nTopics\n\n" + result.topics());

    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

Vedi l'[esempio completo](#) su GitHub.

## Sottoscrizione di un endpoint a un argomento

Dopo aver creato un argomento, puoi configurare quali canali di comunicazione saranno endpoint per tale argomento. I messaggi vengono distribuiti a questi endpoint dopo essere stati ricevuti da Amazon SNS.

Per configurare un canale di comunicazione come endpoint per un argomento, sottoscrivi tale endpoint all'argomento. Per iniziare, costruisci un [SubscribeRequest](#) oggetto. Specificate il canale di comunicazione (ad esempio, `lambda oemail`) come `protocol()`. Imposta la `endpoint()` posizione di output pertinente (ad esempio, l'ARN di una Lambda funzione o un indirizzo e-mail), quindi imposta l'ARN dell'argomento a cui desideri iscriverti come `topicArn()`. Inviare l'oggetto della richiesta a Amazon SNS utilizzando il `subscribe()` metodo di `SnsClient`. È possibile acquisire il risultato di questa richiesta come [SubscribeResponse](#) oggetto.

Il frammento di codice seguente mostra come sottoscrivere un indirizzo e-mail a un argomento.

### Importazioni

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.model.SubscribeRequest;
import software.amazon.awssdk.services.sns.model.SubscribeResponse;
```

### Codice

```
public static void subEmail(SnsClient snsClient, String topicArn, String email) {

    try {
        SubscribeRequest request = SubscribeRequest.builder()
            .protocol("email")
            .endpoint(email)
            .returnSubscriptionArn(true)
            .topicArn(topicArn)
            .build();

        SubscribeResponse result = snsClient.subscribe(request);
        System.out.println("Subscription ARN: " + result.subscriptionArn() + "\n\n"
            + "Status is " + result.sdkHttpResponse().statusCode());
    }
}
```



```
    } catch (SnsException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

Vedi l'[esempio completo](#) su GitHub.

## Pubblicazione di un messaggio in un argomento

Dopo avere configurato un argomento e uno o più endpoint, puoi pubblicare un messaggio. Per iniziare, costruisci un [PublishRequest](#) oggetto. Specifica `message()` per l'invio e l'ARN dell'argomento (`topicArn()`) a cui inviarlo. Quindi, invia l'oggetto richiesto a Amazon SNS utilizzando il metodo `publish()` di `SnsClient`. Puoi acquisire il risultato di questa richiesta come [PublishResponse](#) oggetto.

### Importazioni

```
import software.amazon.awssdk.regions.Region;  
import software.amazon.awssdk.services.sns.SnsClient;  
import software.amazon.awssdk.services.sns.model.PublishRequest;  
import software.amazon.awssdk.services.sns.model.PublishResponse;  
import software.amazon.awssdk.services.sns.model.SnsException;
```

### Codice

```
public static void pubTopic(SnsClient snsClient, String message, String topicArn) {  
  
    try {  
        PublishRequest request = PublishRequest.builder()  
            .message(message)  
            .topicArn(topicArn)  
            .build();  
  
        PublishResponse result = snsClient.publish(request);  
        System.out.println(result.messageId() + " Message sent. Status is " +  
result.sdkHttpResponse().statusCode());  
  
    } catch (SnsException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

```
}
```

Vedi l'[esempio completo](#) su GitHub.

## Annullamento della sottoscrizione di un endpoint da un argomento

Puoi rimuovere i canali di comunicazione configurati come endpoint per un argomento. Dopo aver eseguito questa operazione, l'argomento stesso continua a esistere e distribuire i messaggi a qualsiasi altro endpoint configurato per tale argomento.

Per rimuovere un canale di comunicazione come endpoint per un argomento, annulla la sottoscrizione dell'endpoint dall'argomento. Per iniziare, crea un [UnsubscribeRequest](#) oggetto e imposta l'ARN dell'argomento a cui desideri annullare l'iscrizione come `subscriptionArn()`. Quindi invia l'oggetto della richiesta a SNS utilizzando il metodo `unsubscribe()` di `SnsClient`. Puoi acquisire il risultato di questa richiesta come oggetto [UnsubscribeResponse](#).

### Importazioni

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.model.UnsubscribeRequest;
import software.amazon.awssdk.services.sns.model.UnsubscribeResponse;
```

### Codice

```
public static void unSub(SnsClient snsClient, String subscriptionArn) {

    try {
        UnsubscribeRequest request = UnsubscribeRequest.builder()
            .subscriptionArn(subscriptionArn)
            .build();

        UnsubscribeResponse result = snsClient.unsubscribe(request);

        System.out.println("\n\nStatus was " +
result.sdkHttpResponse().statusCode()
            + "\n\nSubscription was removed for " + request.subscriptionArn());

    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

```
    }  
}
```

Vedi l'[esempio completo](#) su GitHub.

## Eliminazione di un argomento

Per eliminare un Amazon SNS argomento, crea prima un [DeleteTopicRequest](#) oggetto con l'ARN dell'argomento impostato come `topicArn()` metodo nel generatore. Quindi, invia l'oggetto della richiesta a Amazon SNS utilizzando il metodo `deleteTopic()` di `SnsClient`. È possibile acquisire il risultato di questa richiesta come [DeleteTopicResponse](#) oggetto, come dimostrato nel seguente frammento di codice.

### Importazioni

```
import software.amazon.awssdk.regions.Region;  
import software.amazon.awssdk.services.sns.SnsClient;  
import software.amazon.awssdk.services.sns.model.DeleteTopicRequest;  
import software.amazon.awssdk.services.sns.model.DeleteTopicResponse;  
import software.amazon.awssdk.services.sns.model.SnsException;
```

### Codice

```
public static void deleteSNSTopic(SnsClient snsClient, String topicArn ) {  
  
    try {  
        DeleteTopicRequest request = DeleteTopicRequest.builder()  
            .topicArn(topicArn)  
            .build();  
  
        DeleteTopicResponse result = snsClient.deleteTopic(request);  
        System.out.println("\n\nStatus was " +  
result.sdkHttpResponse().statusCode());  
  
    } catch (SnsException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

Vedi l'[esempio completo](#) su GitHub

Per ulteriori informazioni, consulta la [Guida per gli sviluppatori di Amazon Simple Notification Service](#).

## Lavora con Amazon Simple Queue Service

Questa sezione fornisce esempi di programmazione [Amazon Simple Queue Service](#) con AWS SDK for Java 2.x.

Gli esempi seguenti includono solo il codice necessario per dimostrare ciascuna tecnica. Il [codice di esempio completo è disponibile su GitHub](#). Da qui puoi scaricare un singolo file sorgente o clonare l'archivio localmente per ottenere tutti gli esempi da creare ed eseguire.

### Argomenti

- [Lavora con le code di Amazon Simple Queue Service messaggi](#)
- [Inviare, ricevere ed eliminare Amazon Simple Queue Service messaggi](#)

## Lavora con le code di Amazon Simple Queue Service messaggi

Una coda di messaggi è il container logico utilizzato per l'invio di messaggi in modo affidabile in Amazon Simple Queue Service. Sono disponibili due tipi di code: standard e first-in, first-out (FIFO). Per ulteriori informazioni sulle code e sulle differenze tra questi tipi, consulta la Guida per gli [Amazon Simple Queue Servicesviluppatori](#).

In questo argomento viene descritto come creare, elencare, eliminare e ottenere l'URL di una coda Amazon Simple Queue Service utilizzando AWS SDK for Java.

La `sqsClient` variabile utilizzata negli esempi seguenti può essere creata dal seguente frammento.

```
SqsClient sqsClient = SqsClient.create();
```

[Quando si crea un SqsClient utilizzando il create\(\) metodo statico, l'SDK configura la regione utilizzando la catena di provider di aree predefinita e le credenziali utilizzando la catena di provider di credenziali predefinita.](#)

### Crea una coda

Utilizzate il `SqsClient`'s `createQueue` metodo e fornite un [CreateQueueRequest](#) oggetto che descriva i parametri della coda come mostrato nel seguente frammento di codice.

### Importazioni

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.*;
import java.util.List;
```

## Codice

```
        CreateQueueRequest createQueueRequest = CreateQueueRequest.builder()
            .queueName(queueName)
            .build();

        sqsClient.createQueue(createQueueRequest);
```

Vedi l'esempio [completo](#) su GitHub

## Elencare code

Per elencare le Amazon Simple Queue Service code relative al tuo account, chiama il `SqsClient`'s `listQueues` metodo con un [ListQueuesRequest](#) oggetto.

Quando utilizzate il modulo del [listQueues](#) metodo che non accetta parametri, il servizio restituisce tutte le code, fino a 1.000 code.

È possibile fornire un prefisso per il nome della coda all'[ListQueuesRequest](#) oggetto per limitare i risultati alle code che corrispondono a tale prefisso, come illustrato nel codice seguente.

## Importazioni

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.*;
import java.util.List;
```

## Codice

```
        String prefix = "que";

        try {
            ListQueuesRequest listQueuesRequest =
                ListQueuesRequest.builder().queueNamePrefix(prefix).build();
            ListQueuesResponse listQueuesResponse =
                sqsClient.listQueues(listQueuesRequest);
```

```
        for (String url : listQueuesResponse.queueUrls()) {
            System.out.println(url);
        }

    } catch (SqsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

[Guarda l'esempio completo su GitHub](#)

## Ottenere l'URL di una coda

Il codice seguente mostra come ottenere l'URL di una coda chiamando il `SqsClient`'s `getQueueUrl` metodo con un [GetQueueUrlRequest](#) oggetto.

### Importazioni

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.*;
import java.util.List;
```

### Codice

```
        GetQueueUrlResponse getQueueUrlResponse =

sqsClient.getQueueUrl(GetQueueUrlRequest.builder().queueName(queueName).build());
        String queueUrl = getQueueUrlResponse.queueUrl();
        return queueUrl;
```

[Guarda l'esempio completo su GitHub](#)

## Elimina una coda

Fornisci l'[URL](#) della coda all'[DeleteQueueRequest](#) oggetto. Quindi chiamate il `SqsClient`'s `deleteQueue` metodo per eliminare una coda come mostrato nel codice seguente.

### Importazioni

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sqs.SqsClient;
```

```
import software.amazon.awssdk.services.sqs.model.*;
import java.util.List;
```

## Codice

```
public static void deleteSQSQueue(SqsClient sqsClient, String queueName) {

    try {

        GetQueueUrlRequest getQueueRequest = GetQueueUrlRequest.builder()
            .queueName(queueName)
            .build();

        String queueUrl = sqsClient.getQueueUrl(getQueueRequest).queueUrl();

        DeleteQueueRequest deleteQueueRequest = DeleteQueueRequest.builder()
            .queueUrl(queueUrl)
            .build();

        sqsClient.deleteQueue(deleteQueueRequest);

    } catch (SqsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

Guarda l'[esempio completo](#) su. GitHub

## Ulteriori informazioni

- [CreateQueue](#) nell'Amazon Simple Queue Service API Reference
- [GetQueueUrl](#) nell'Amazon Simple Queue Service API Reference
- [ListQueues](#) nell'Amazon Simple Queue Service API Reference
- [DeleteQueue](#) nell'Amazon Simple Queue Service API Reference

## Inviare, ricevere ed eliminare Amazon Simple Queue Service messaggi

Un messaggio è una parte di dati che possono essere inviati e ricevuti da componenti distribuiti. I messaggi vengono sempre distribuiti tramite una [coda SQS](#).

La `SqsClient` variabile utilizzata negli esempi seguenti può essere creata dal seguente frammento.

```
SqsClient sqsClient = SqsClient.create();
```

[Quando si crea un `SqsClient` utilizzando il `create\(\)` metodo statico, l'SDK configura la regione utilizzando la catena di provider di aree predefinita e le credenziali utilizzando la catena di provider di credenziali predefinita.](#)

## Invio di un messaggio

Aggiungi un singolo messaggio a una Amazon Simple Queue Service coda chiamando il metodo client. `SqsClient sendMessage` Fornisci un [SendMessageRequest](#) oggetto che contenga l'[URL](#) della coda, il corpo del messaggio e il valore di ritardo opzionale (in secondi).

### Importazioni

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.*;
import java.util.List;
```

### Codice

```
sqsClient.sendMessage(SendMessageRequest.builder()
    .queueUrl(queueUrl)
    .messageBody("Hello world!")
    .delaySeconds(10)
    .build());

sqsClient.sendMessage(sendMsgRequest);
```

## Invia più messaggi in una richiesta

Invia più messaggi in una singola richiesta utilizzando il metodo `SqsClient` di `sendMessageBatch`. Questo metodo richiede un file [SendMessageBatchRequest](#) che contenga l'URL della coda e un elenco di messaggi da inviare. (Ogni messaggio è un [SendMessageBatchRequestEntry](#).) Puoi anche ritardare l'invio di un messaggio specifico impostando un valore di ritardo nel messaggio.

### Importazioni

```
import software.amazon.awssdk.regions.Region;
```



```
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.*;
import java.util.List;
```

## Codice

```
SendMessageBatchRequest sendMessageBatchRequest =
SendMessageBatchRequest.builder()
    .queueUrl(queueUrl)

    .entries(SendMessageBatchRequestEntry.builder().id("id1").messageBody("Hello from msg
1").build(),

SendMessageBatchRequestEntry.builder().id("id2").messageBody("msg
2").delaySeconds(10).build())
    .build();
sqsClient.sendMessageBatch(sendMessageBatchRequest);
```

Guarda l'[esempio completo](#) su GitHub.

## Recuperare messaggi

Recupera eventuali messaggi che si trovano attualmente nella coda chiamando il metodo `SqsClient` di `receiveMessage`. Questo metodo accetta un file [ReceiveMessageRequest](#) che contiene l'URL della coda. Puoi specificare il numero massimo di messaggi da restituire. I messaggi vengono restituiti come un elenco di oggetti [Message](#).

## Importazioni

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.*;
import java.util.List;
```

## Codice

```
try {
    ReceiveMessageRequest receiveMessageRequest =
ReceiveMessageRequest.builder()
    .queueUrl(queueUrl)
    .numberOfMessages(5)
    .build();
```

```
        List<Message> messages =
sqsClient.receiveMessage(receiveMessageRequest).messages();
        return messages;
    } catch (SqsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return null;
}
```

Guarda l'[esempio completo](#) su. GitHub

## Eliminare un messaggio dopo la ricezione

Dopo aver ricevuto un messaggio e averne elaborato il contenuto, eliminalo dalla coda inviando al metodo l'handle di ricezione e l'URL di coda del SqsClient 's [deleteMessage](#) messaggio.

### Importazioni

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.*;
import java.util.List;
```

### Codice

```
    try {
        for (Message message : messages) {
            DeleteMessageRequest deleteMessageRequest =
DeleteMessageRequest.builder()
                .queueUrl(queueUrl)
                .receiptHandle(message.receiptHandle())
                .build();
            sqsClient.deleteMessage(deleteMessageRequest);
        }
    }
```

Guarda l'[esempio completo](#) su. GitHub

## Ulteriori informazioni

- [Come funzionano Amazon Simple Queue Service le code](#) nella Guida per gli Amazon Simple Queue Service sviluppatori
- [SendMessage](#) nell'Amazon Simple Queue Service API Reference

- [SendMessageBatch](#) nell'Amazon Simple Queue Service API Reference
- [ReceiveMessage](#) nell'Amazon Simple Queue Service API Reference
- [DeleteMessage](#) nell'Amazon Simple Queue Service API Reference

## Utilizzo delle Amazon Transcribe

L'esempio seguente mostra come funziona lo streaming bidirezionale con Amazon Transcribe. Lo streaming bidirezionale implica che ci sia un flusso di dati verso il servizio e in direzione opposta in tempo reale. L'esempio utilizza la trascrizione dello streaming Amazon Transcribe per inviare un flusso audio e ricevere un flusso di testo trascritto in tempo reale.

Per ulteriori informazioni [su questa funzionalità, consulta Streaming Transcription](#) nella Guida per gli Amazon Transcribe sviluppatori.

Per [iniziare a utilizzare](#), consulta la sezione Guida introduttiva nella Guida per gli Amazon Transcribe sviluppatori. Amazon Transcribe

## Configura il microfono

Questo codice usa il pacchetto `javax.sound.sampled` per lo streaming audio da un dispositivo di input.

### Codice

```
import javax.sound.sampled.AudioFormat;
import javax.sound.sampled.AudioSystem;
import javax.sound.sampled.DataLine;
import javax.sound.sampled.TargetDataLine;

public class Microphone {

    public static TargetDataLine get() throws Exception {
        AudioFormat format = new AudioFormat(16000, 16, 1, true, false);
        DataLine.Info datalineInfo = new DataLine.Info(TargetDataLine.class, format);

        TargetDataLine dataLine = (TargetDataLine) AudioSystem.getLine(datalineInfo);
        dataLine.open(format);

        return dataLine;
    }
}
```

Guarda l'[esempio completo](#) su GitHub.

## Crea un editore

Questo codice implementa un publisher che pubblica dati audio dal flusso audio Amazon Transcribe.

### Codice

```
package com.amazonaws.transcribe;

import java.io.IOException;
import java.io.InputStream;
import java.io.UncheckedIOException;
import java.nio.ByteBuffer;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;
import java.util.concurrent.atomic.AtomicLong;
import org.reactivestreams.Publisher;
import org.reactivestreams.Subscriber;
import org.reactivestreams.Subscription;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.services.transcribestreaming.model.AudioEvent;
import software.amazon.awssdk.services.transcribestreaming.model.AudioStream;
import
    software.amazon.awssdk.services.transcribestreaming.model.TranscribeStreamingException;

public class AudioStreamPublisher implements Publisher<AudioStream> {
    private final InputStream inputStream;

    public AudioStreamPublisher(InputStream inputStream) {
        this.inputStream = inputStream;
    }

    @Override
    public void subscribe(Subscriber<? super AudioStream> s) {
        s.onSubscribe(new SubscriptionImpl(s, inputStream));
    }

    private class SubscriptionImpl implements Subscription {
        private static final int CHUNK_SIZE_IN_BYTES = 1024 * 1;
        private ExecutorService executor = Executors.newFixedThreadPool(1);
        private AtomicLong demand = new AtomicLong(0);
    }
}
```

```
private final Subscriber<? super AudioStream> subscriber;
private final InputStream inputStream;

private SubscriptionImpl(Subscriber<? super AudioStream> s, InputStream
inputStream) {
    this.subscriber = s;
    this.inputStream = inputStream;
}

@Override
public void request(long n) {
    if (n <= 0) {
        subscriber.onError(new IllegalArgumentException("Demand must be
positive"));
    }

    demand.getAndAdd(n);

    executor.submit(() -> {
        try {
            do {
                ByteBuffer audioBuffer = getNextEvent();
                if (audioBuffer.remaining() > 0) {
                    AudioEvent audioEvent = audioEventFromBuffer(audioBuffer);
                    subscriber.onNext(audioEvent);
                } else {
                    subscriber.onComplete();
                    break;
                }
            } while (demand.decrementAndGet() > 0);
        } catch (TranscribeStreamingException e) {
            subscriber.onError(e);
        }
    });
}

@Override
public void cancel() {

}

private ByteBuffer getNextEvent() {
    ByteBuffer audioBuffer;
    byte[] audioBytes = new byte[CHUNK_SIZE_IN_BYTES];
```

```
int len = 0;
try {
    len = inputStream.read(audioBytes);

    if (len <= 0) {
        audioBuffer = ByteBuffer.allocate(0);
    } else {
        audioBuffer = ByteBuffer.wrap(audioBytes, 0, len);
    }
} catch (IOException e) {
    throw new UncheckedIOException(e);
}

return audioBuffer;
}

private AudioEvent audioEventFromBuffer(ByteBuffer bb) {
    return AudioEvent.builder()
        .audioChunk(SdkBytes.fromByteBuffer(bb))
        .build();
}
}
```

Guarda l'[esempio completo](#) su GitHub.

## Crea il client e avvia lo streaming

Nel metodo principale, è necessario creare un oggetto richiesta, avviare il flusso di input audio e creare un'istanza per il publisher con l'input audio.

È inoltre necessario creare un [StartStreamTranscriptionResponseHandler](#) file per specificare come gestire la risposta da Amazon Transcribe.

Quindi, usa il `startStreamTranscription` metodo `TranscribeStreamingAsyncClient`'s per avviare lo streaming bidirezionale.

### Importazioni

```
import javax.sound.sampled.AudioFormat;
import javax.sound.sampled.AudioSystem;
import javax.sound.sampled.DataLine;
```

```

import javax.sound.sampled.TargetDataLine;
import javax.sound.sampled.AudioInputStream;
import software.amazon.awssdk.regions.Region;
import
    software.amazon.awssdk.services.transcribestreaming.TranscribeStreamingAsyncClient;
import
    software.amazon.awssdk.services.transcribestreaming.model.TranscribeStreamingException ;
import
    software.amazon.awssdk.services.transcribestreaming.model.StartStreamTranscriptionRequest;
import software.amazon.awssdk.services.transcribestreaming.model.MediaEncoding;
import software.amazon.awssdk.services.transcribestreaming.model.LanguageCode;
import
    software.amazon.awssdk.services.transcribestreaming.model.StartStreamTranscriptionResponseHandler;
import software.amazon.awssdk.services.transcribestreaming.model.TranscriptEvent;

```

## Codice

```

    public static void convertAudio(TranscribeStreamingAsyncClient client) throws
    Exception {

        try {

            StartStreamTranscriptionRequest request =
            StartStreamTranscriptionRequest.builder()
                .mediaEncoding(MediaEncoding.PCM)
                .languageCode(LanguageCode.EN_US)
                .mediaSampleRateHertz(16_000).build();

            TargetDataLine mic = Microphone.get();
            mic.start();

            AudioStreamPublisher publisher = new AudioStreamPublisher(new
            AudioInputStream(mic));

            StartStreamTranscriptionResponseHandler response =
                StartStreamTranscriptionResponseHandler.builder().subscriber(e -> {
                    TranscriptEvent event = (TranscriptEvent) e;
                    event.transcript().results().forEach(r ->
                    r.alternatives().forEach(a -> System.out.println(a.transcript())));
                }).build();

            // Keeps Streaming until you end the Java program
            client.startStreamTranscription(request, publisher, response);

```

```
    } catch (TranscribeStreamingException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

Guarda l'[esempio completo](#) su GitHub

## Ulteriori informazioni

- [Come funziona](#) nella Guida per gli Amazon Transcribe sviluppatori.
- [Guida introduttiva allo streaming audio](#) nella Guida per Amazon Transcribe gli sviluppatori.



## Esempi di codice SDK for Java 2.x

Gli esempi di codice riportati in questo argomento mostrano come utilizzare AWS SDK for Java 2.x with AWS.

Le operazioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le operazioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati e negli esempi tra servizi.

Scenari: esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio.

Esempi cross-service: applicazioni di esempio che funzionano su più servizi Servizi AWS.

### Esempi

- [Azioni e scenari che utilizzano SDK for Java 2.x](#)
- [Esempi cross-service che utilizzano SDK for Java 2.x](#)

## Azioni e scenari che utilizzano SDK for Java 2.x

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando with. AWS SDK for Java 2.x Servizi AWS

Le operazioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le operazioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati e negli esempi tra servizi.

Scenari: esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio.

### Servizi

- [Esempi di API Gateway che utilizzano SDK for Java 2.x](#)
- [Esempi di Application Auto Scaling con SDK for Java 2.x](#)
- [Esempi di Application Recovery Controller che utilizzano SDK for Java 2.x](#)
- [Esempi di Aurora con SDK for Java 2.x](#)
- [Esempi di Auto Scaling con SDK for Java 2.x](#)

- [Esempi di Amazon Bedrock con SDK for Java 2.x](#)
- [Esempi di Amazon Bedrock Runtime con SDK for Java 2.x](#)
- [CloudFront esempi che utilizzano SDK for Java 2.x](#)
- [CloudWatch esempi che utilizzano SDK for Java 2.x](#)
- [CloudWatch Esempi di eventi che utilizzano SDK for Java 2.x](#)
- [CloudWatch Esempi di log utilizzando SDK for Java 2.x](#)
- [Esempi di identità di Amazon Cognito con SDK for Java 2.x](#)
- [Esempi di Amazon Cognito Identity Provider che utilizzano SDK for Java 2.x](#)
- [Esempi di Amazon Comprehend con SDK for Java 2.x](#)
- [Esempi di DynamoDB con SDK for Java 2.x](#)
- [Esempi di Amazon EC2 che utilizzano SDK for Java 2.x](#)
- [Esempi di Amazon ECS con SDK for Java 2.x](#)
- [Elastic Load Balancing - Esempi della versione 2 che utilizzano SDK for Java 2.x](#)
- [MediaStore esempi che utilizzano SDK for Java 2.x](#)
- [OpenSearch Esempi di servizi che utilizzano SDK for Java 2.x](#)
- [EventBridge esempi che utilizzano SDK for Java 2.x](#)
- [Esempi di previsione con SDK for Java 2.x](#)
- [AWS Glue esempi che utilizzano SDK for Java 2.x](#)
- [HealthImaging esempi che utilizzano SDK for Java 2.x](#)
- [Esempi IAM che utilizzano SDK for Java 2.x](#)
- [AWS IoT esempi che utilizzano SDK for Java 2.x](#)
- [AWS IoT data esempi che utilizzano SDK for Java 2.x](#)
- [Esempi di Amazon Keyspaces con SDK for Java 2.x](#)
- [Esempi di Kinesis con SDK for Java 2.x](#)
- [AWS KMS esempi che utilizzano SDK for Java 2.x](#)
- [Esempi di Lambda con SDK for Java 2.x](#)
- [MediaConvert esempi che utilizzano SDK for Java 2.x](#)
- [Esempi di Migration Hub con SDK for Java 2.x](#)
- [Esempi di Amazon Personalize con SDK for Java 2.x](#)
- [Esempi di Amazon Personalize Events con SDK for Java 2.x](#)

- [Esempi di Amazon Personalize Runtime con SDK for Java 2.x](#)
- [Esempi di Amazon Pinpoint con SDK for Java 2.x](#)
- [Esempi di API Amazon Pinpoint SMS e Voice con SDK for Java 2.x](#)
- [Esempi di Amazon Polly con SDK for Java 2.x](#)
- [Esempi di Amazon RDS che utilizzano SDK for Java 2.x](#)
- [Esempi di Amazon Redshift con SDK for Java 2.x](#)
- [Esempi di Amazon Rekognition con SDK for Java 2.x](#)
- [Esempi di registrazione del dominio Route 53 utilizzando SDK for Java 2.x](#)
- [Esempi di Amazon S3 con SDK for Java 2.x](#)
- [Esempi di S3 Glacier con SDK for Java 2.x](#)
- [SageMaker esempi che utilizzano SDK for Java 2.x](#)
- [Esempi di Secrets Manager con SDK for Java 2.x](#)
- [Esempi di Amazon SES con SDK for Java 2.x](#)
- [Esempi di API Amazon SES v2 con SDK for Java 2.x](#)
- [Esempi di Amazon SNS con SDK for Java 2.x](#)
- [Esempi di Amazon SQS con SDK for Java 2.x](#)
- [Esempi di Step Functions con SDK for Java 2.x](#)
- [AWS STS esempi che utilizzano SDK for Java 2.x](#)
- [AWS Support esempi che utilizzano SDK for Java 2.x](#)
- [Esempi di Systems Manager che utilizzano SDK for Java 2.x](#)
- [Esempi di Amazon Textract con SDK for Java 2.x](#)
- [Esempi di Amazon Transcribe con SDK for Java 2.x](#)

## Esempi di API Gateway che utilizzano SDK for Java 2.x

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando AWS SDK for Java 2.x with API Gateway.

Le operazioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le operazioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati e negli esempi tra servizi.

Scenari: esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio.

Ogni esempio include un collegamento a GitHub, dove è possibile trovare istruzioni su come configurare ed eseguire il codice nel contesto.

Argomenti

- [Azioni](#)

Azioni

## CreateDeployment

Il seguente esempio di codice mostra come utilizzare `CreateDeployment`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public static String createNewDeployment(ApiGatewayClient apiGateway, String
restApiId, String stageName) {

    try {
        CreateDeploymentRequest request = CreateDeploymentRequest.builder()
            .restApiId(restApiId)
            .description("Created using the AWS API Gateway Java API")
            .stageName(stageName)
            .build();

        CreateDeploymentResponse response =
apiGateway.createDeployment(request);
        System.out.println("The id of the deployment is " + response.id());
        return response.id();

    } catch (ApiGatewayException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }
}
```

```
        System.exit(1);
    }
    return "";
}
```

- Per i dettagli sull'API, consulta la [CreateDeployment](#) sezione AWS SDK for Java 2.x API Reference.

## CreateRestApi

Il seguente esempio di codice mostra come utilizzare `CreateRestApi`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public static String createAPI(ApiGatewayClient apiGateway, String restApiId,
String restApiName) {

    try {
        CreateRestApiRequest request = CreateRestApiRequest.builder()
            .cloneFrom(restApiId)
            .description("Created using the Gateway Java API")
            .name(restApiName)
            .build();

        CreateRestApiResponse response = apiGateway.createRestApi(request);
        System.out.println("The id of the new api is " + response.id());
        return response.id();

    } catch (ApiGatewayException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
```

- Per i dettagli sull'API, consulta la [CreateRestApi](#) sezione AWS SDK for Java 2.x API Reference.

## DeleteDeployment

Il seguente esempio di codice mostra come utilizzare `DeleteDeployment`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public static void deleteSpecificDeployment(ApiGatewayClient apiGateway, String
restApiId, String deploymentId) {

    try {
        DeleteDeploymentRequest request = DeleteDeploymentRequest.builder()
            .restApiId(restApiId)
            .deploymentId(deploymentId)
            .build();

        apiGateway.deleteDeployment(request);
        System.out.println("Deployment was deleted");

    } catch (ApiGatewayException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Per i dettagli sull'API, consulta la [DeleteDeployment](#) sezione AWS SDK for Java 2.x API Reference.

## DeleteRestApi

Il seguente esempio di codice mostra come utilizzare `DeleteRestApi`.

## SDK per Java 2.x

 Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public static void deleteAPI(ApiGatewayClient apiGateway, String restApiId) {  
  
    try {  
        DeleteRestApiRequest request = DeleteRestApiRequest.builder()  
            .restApiId(restApiId)  
            .build();  
  
        apiGateway.deleteRestApi(request);  
        System.out.println("The API was successfully deleted");  
  
    } catch (ApiGatewayException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

- Per i dettagli sull'API, consulta la [DeleteRestApi](#) sezione AWS SDK for Java 2.x API Reference.

## Esempi di Application Auto Scaling con SDK for Java 2.x

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando AWS SDK for Java 2.x with Application Auto Scaling.

Le operazioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le operazioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati e negli esempi tra servizi.

Scenari: esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio.

Ogni esempio include un collegamento a GitHub, dove è possibile trovare istruzioni su come configurare ed eseguire il codice nel contesto.

## Argomenti

- [Azioni](#)

## Azioni

### DeleteScalingPolicy

Il seguente esempio di codice mostra come utilizzare `DeleteScalingPolicy`.

#### SDK per Java 2.x

#### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;
import
    software.amazon.awssdk.services.applicationautoscaling.ApplicationAutoScalingClient;
import
    software.amazon.awssdk.services.applicationautoscaling.model.ApplicationAutoScalingException;
import
    software.amazon.awssdk.services.applicationautoscaling.model.DeleteScalingPolicyRequest;
import
    software.amazon.awssdk.services.applicationautoscaling.model.DeregisterScalableTargetRequest;
import
    software.amazon.awssdk.services.applicationautoscaling.model.DescribeScalableTargetsRequest;
import
    software.amazon.awssdk.services.applicationautoscaling.model.DescribeScalableTargetsResponse;
import
    software.amazon.awssdk.services.applicationautoscaling.model.DescribeScalingPoliciesRequest;
import
    software.amazon.awssdk.services.applicationautoscaling.model.DescribeScalingPoliciesResponse;
import
    software.amazon.awssdk.services.applicationautoscaling.model.ScalableDimension;
import
    software.amazon.awssdk.services.applicationautoscaling.model.ServiceNamespace;

/**
```



```
* Before running this Java V2 code example, set up your development environment,  
including your credentials.  
*  
* For more information, see the following documentation topic:  
*  
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html  
*/
```

```
public class DisableDynamoDBAutoScaling {  
    public static void main(String[] args) {  
        final String usage = ""  
  
            Usage:  
            <tableId> <policyName>\s  
  
            Where:  
            tableId - The table Id value (for example, table/Music).\s  
            policyName - The name of the policy (for example, $Music5-scaling-  
policy).  
  
            "";  
        if (args.length != 2) {  
            System.out.println(usage);  
            System.exit(1);  
        }  
  
        ApplicationAutoScalingClient appAutoScalingClient =  
ApplicationAutoScalingClient.builder()  
            .region(Region.US_EAST_1)  
            .build();  
  
        ServiceNamespace ns = ServiceNamespace.DYNAMODB;  
        ScalableDimension tableWCUs =  
ScalableDimension.DYNAMODB_TABLE_WRITE_CAPACITY_UNITS;  
        String tableId = args[0];  
        String policyName = args[1];  
  
        deletePolicy(appAutoScalingClient, policyName, tableWCUs, ns, tableId);  
        verifyScalingPolicies(appAutoScalingClient, tableId, ns, tableWCUs);  
        deregisterScalableTarget(appAutoScalingClient, tableId, ns, tableWCUs);  
        verifyTarget(appAutoScalingClient, tableId, ns, tableWCUs);  
    }  
}
```

```
public static void deletePolicy(ApplicationAutoScalingClient
appAutoScalingClient, String policyName, ScalableDimension tableWCUs,
ServiceNamespace ns, String tableId) {
    try {
        DeleteScalingPolicyRequest delSPRequest =
DeleteScalingPolicyRequest.builder()
        .policyName(policyName)
        .scalableDimension(tableWCUs)
        .serviceNamespace(ns)
        .resourceId(tableId)
        .build();

        appAutoScalingClient.deleteScalingPolicy(delSPRequest);
        System.out.println(policyName + " was deleted successfully.");

    } catch (ApplicationAutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }
}

// Verify that the scaling policy was deleted
public static void verifyScalingPolicies(ApplicationAutoScalingClient
appAutoScalingClient, String tableId, ServiceNamespace ns, ScalableDimension
tableWCUs) {
    DescribeScalingPoliciesRequest dscRequest =
DescribeScalingPoliciesRequest.builder()
    .scalableDimension(tableWCUs)
    .serviceNamespace(ns)
    .resourceId(tableId)
    .build();

    DescribeScalingPoliciesResponse response =
appAutoScalingClient.describeScalingPolicies(dscRequest);
    System.out.println("DescribeScalableTargets result: ");
    System.out.println(response);
}

public static void deregisterScalableTarget(ApplicationAutoScalingClient
appAutoScalingClient, String tableId, ServiceNamespace ns, ScalableDimension
tableWCUs) {
    try {
        DeregisterScalableTargetRequest targetRequest =
DeregisterScalableTargetRequest.builder()
        .scalableDimension(tableWCUs)
```

```
        .serviceNamespace(ns)
        .resourceId(tableId)
        .build();

    appAutoScalingClient.deregisterScalableTarget(targetRequest);
    System.out.println("The scalable target was deregistered.");

} catch (ApplicationAutoScalingException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
}
}

public static void verifyTarget(ApplicationAutoScalingClient
appAutoScalingClient, String tableId, ServiceNamespace ns, ScalableDimension
tableWCUs) {
    DescribeScalableTargetsRequest dscRequest =
DescribeScalableTargetsRequest.builder()
        .scalableDimension(tableWCUs)
        .serviceNamespace(ns)
        .resourceIds(tableId)
        .build();

    DescribeScalableTargetsResponse response =
appAutoScalingClient.describeScalableTargets(dscRequest);
    System.out.println("DescribeScalableTargets result: ");
    System.out.println(response);
}
}
```

- Per i dettagli sull'API, consulta la [DeleteScalingPolicy](#) sezione AWS SDK for Java 2.x API Reference.

## RegisterScalableTarget

Il seguente esempio di codice mostra come utilizzare `RegisterScalableTarget`.

## SDK per Java 2.x

 Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;
import
    software.amazon.awssdk.services.applicationautoscaling.ApplicationAutoScalingClient;
import
    software.amazon.awssdk.services.applicationautoscaling.model.ApplicationAutoScalingException;
import
    software.amazon.awssdk.services.applicationautoscaling.model.DescribeScalableTargetsRequest;
import
    software.amazon.awssdk.services.applicationautoscaling.model.DescribeScalableTargetsResponse;
import
    software.amazon.awssdk.services.applicationautoscaling.model.DescribeScalingPoliciesRequest;
import
    software.amazon.awssdk.services.applicationautoscaling.model.DescribeScalingPoliciesResponse;
import software.amazon.awssdk.services.applicationautoscaling.model.PolicyType;
import
    software.amazon.awssdk.services.applicationautoscaling.model.PredefinedMetricSpecification;
import
    software.amazon.awssdk.services.applicationautoscaling.model.PutScalingPolicyRequest;
import
    software.amazon.awssdk.services.applicationautoscaling.model.RegisterScalableTargetRequest;
import software.amazon.awssdk.services.applicationautoscaling.model.ScalingPolicy;
import
    software.amazon.awssdk.services.applicationautoscaling.model.ServiceNamespace;
import
    software.amazon.awssdk.services.applicationautoscaling.model.ScalableDimension;
import software.amazon.awssdk.services.applicationautoscaling.model.MetricType;
import
    software.amazon.awssdk.services.applicationautoscaling.model.TargetTrackingScalingPolicyConfiguration;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development environment,
 * including your credentials.
 */
```

```
* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class EnableDynamoDBAutoscaling {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <tableId> <roleARN> <policyName>\s

            Where:
                tableId - The table Id value (for example, table/Music).
                roleARN - The ARN of the role that has ApplicationAutoScaling
permissions.
                policyName - The name of the policy to create.

            """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        System.out.println("This example registers an Amazon DynamoDB table, which
is the resource to scale.");
        String tableId = args[0];
        String roleARN = args[1];
        String policyName = args[2];
        ServiceNamespace ns = ServiceNamespace.DYNAMODB;
        ScalableDimension tableWCUs =
ScalableDimension.DYNAMODB_TABLE_WRITE_CAPACITY_UNITS;
        ApplicationAutoScalingClient appAutoScalingClient =
ApplicationAutoScalingClient.builder()
            .region(Region.US_EAST_1)
            .build();

        registerScalableTarget(appAutoScalingClient, tableId, roleARN, ns,
tableWCUs);
        verifyTarget(appAutoScalingClient, tableId, ns, tableWCUs);
        configureScalingPolicy(appAutoScalingClient, tableId, ns, tableWCUs,
policyName);
    }
}
```

```
public static void registerScalableTarget(ApplicationAutoScalingClient
appAutoScalingClient, String tableId, String roleARN, ServiceNamespace ns,
ScalableDimension tableWCUs) {
    try {
        RegisterScalableTargetRequest targetRequest =
RegisterScalableTargetRequest.builder()
            .serviceNamespace(ns)
            .scalableDimension(tableWCUs)
            .resourceId(tableId)
            .roleARN(roleARN)
            .minCapacity(5)
            .maxCapacity(10)
            .build();

        appAutoScalingClient.registerScalableTarget(targetRequest);
        System.out.println("You have registered " + tableId);

    } catch (ApplicationAutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }
}

// Verify that the target was created.
public static void verifyTarget(ApplicationAutoScalingClient
appAutoScalingClient, String tableId, ServiceNamespace ns, ScalableDimension
tableWCUs) {
    DescribeScalableTargetsRequest dscRequest =
DescribeScalableTargetsRequest.builder()
        .scalableDimension(tableWCUs)
        .serviceNamespace(ns)
        .resourceIds(tableId)
        .build();

    DescribeScalableTargetsResponse response =
appAutoScalingClient.describeScalableTargets(dscRequest);
    System.out.println("DescribeScalableTargets result: ");
    System.out.println(response);
}

// Configure a scaling policy.
public static void configureScalingPolicy(ApplicationAutoScalingClient
appAutoScalingClient, String tableId, ServiceNamespace ns, ScalableDimension
tableWCUs, String policyName) {
    // Check if the policy exists before creating a new one.
```

```
DescribeScalingPoliciesResponse describeScalingPoliciesResponse =
appAutoScalingClient.describeScalingPolicies(DescribeScalingPoliciesRequest.builder()
    .serviceNamespace(ns)
    .resourceId(tableId)
    .scalableDimension(tableWCUs)
    .build());

if (!describeScalingPoliciesResponse.scalingPolicies().isEmpty()) {
    // If policies exist, consider updating an existing policy instead of
    creating a new one.
    System.out.println("Policy already exists. Consider updating it
instead.");
    List<ScalingPolicy> polList =
describeScalingPoliciesResponse.scalingPolicies();
    for (ScalingPolicy pol : polList) {
        System.out.println("Policy name:" +pol.policyName());
    }
} else {
    // If no policies exist, proceed with creating a new policy.
    PredefinedMetricSpecification specification =
PredefinedMetricSpecification.builder()

.predefinedMetricType(MetricType.DYNAMO_DB_WRITE_CAPACITY_UTILIZATION)
    .build();

    TargetTrackingScalingPolicyConfiguration policyConfiguration =
TargetTrackingScalingPolicyConfiguration.builder()
    .predefinedMetricSpecification(specification)
    .targetValue(50.0)
    .scaleInCooldown(60)
    .scaleOutCooldown(60)
    .build();

    PutScalingPolicyRequest putScalingPolicyRequest =
PutScalingPolicyRequest.builder()
    .targetTrackingScalingPolicyConfiguration(policyConfiguration)
    .serviceNamespace(ns)
    .scalableDimension(tableWCUs)
    .resourceId(tableId)
    .policyName(policyName)
    .policyType(PolicyType.TARGET_TRACKING_SCALING)
    .build();

    try {
```

```
        appAutoScalingClient.putScalingPolicy(putScalingPolicyRequest);
        System.out.println("You have successfully created a scaling policy
for an Application Auto Scaling scalable target");
    } catch (ApplicationAutoScalingException e) {
        System.err.println("Error: " + e.awsErrorDetails().errorMessage());
    }
}
}
```

- Per i dettagli sull'API, consulta la [RegisterScalableTarget](#) sezione AWS SDK for Java 2.x API Reference.

## Esempi di Application Recovery Controller che utilizzano SDK for Java 2.x

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando Application Recovery Controller. AWS SDK for Java 2.x

Le operazioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le operazioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati e negli esempi tra servizi.

Scenari: esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio.

Ogni esempio include un collegamento a GitHub, dove è possibile trovare istruzioni su come configurare ed eseguire il codice nel contesto.

### Argomenti

- [Azioni](#)

### Azioni

#### **GetRoutingControlState**

Il seguente esempio di codice mostra come utilizzare `GetRoutingControlState`.



## SDK per Java 2.x

 Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public static GetRoutingControlStateResponse
getRoutingControlState(List<ClusterEndpoint> clusterEndpoints,
    String routingControlArn) {
    // As a best practice, we recommend choosing a random cluster endpoint to
    get or
    // set routing control states.
    // For more information, see
    // https://docs.aws.amazon.com/r53recovery/latest/dg/route53-arc-best-
    practices.html#route53-arc-best-practices.regional
    Collections.shuffle(clusterEndpoints);
    for (ClusterEndpoint clusterEndpoint : clusterEndpoints) {
        try {
            System.out.println(clusterEndpoint);
            Route53RecoveryClusterClient client =
Route53RecoveryClusterClient.builder()
                .endpointOverride(URI.create(clusterEndpoint.endpoint()))
                .region(Region.of(clusterEndpoint.region())).build();
            return client.getRoutingControlState(
                GetRoutingControlStateRequest.builder()
                    .routingControlArn(routingControlArn).build());
        } catch (Exception exception) {
            System.out.println(exception);
        }
    }
    return null;
}
```

- Per i dettagli sull'API, consulta la [GetRoutingControlState](#) sezione AWS SDK for Java 2.x API Reference.

## UpdateRoutingControlState

Il seguente esempio di codice mostra come utilizzare `UpdateRoutingControlState`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public static UpdateRoutingControlStateResponse
updateRoutingControlState(List<ClusterEndpoint> clusterEndpoints,
    String routingControlArn,
    String routingControlState) {
    // As a best practice, we recommend choosing a random cluster endpoint to
    get or
    // set routing control states.
    // For more information, see
    // https://docs.aws.amazon.com/r53recovery/latest/dg/route53-arc-best-
    practices.html#route53-arc-best-practices.regional
    Collections.shuffle(clusterEndpoints);
    for (ClusterEndpoint clusterEndpoint : clusterEndpoints) {
        try {
            System.out.println(clusterEndpoint);
            Route53RecoveryClusterClient client =
Route53RecoveryClusterClient.builder()
                .endpointOverride(URI.create(clusterEndpoint.endpoint()))
                .region(Region.of(clusterEndpoint.region()))
                .build();
            return client.updateRoutingControlState(
                UpdateRoutingControlStateRequest.builder()

.routingControlArn(routingControlArn).routingControlState(routingControlState).build());
        } catch (Exception exception) {
            System.out.println(exception);
        }
    }
    return null;
}
```

- Per i dettagli sull'API, consulta la [UpdateRoutingControlState](#) sezione AWS SDK for Java 2.x API Reference.

## Esempi di Aurora con SDK for Java 2.x

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando AWS SDK for Java 2.x con Aurora.

Le operazioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le operazioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati e negli esempi tra servizi.

Scenari: esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio.

Ogni esempio include un collegamento a GitHub, dove è possibile trovare istruzioni su come configurare ed eseguire il codice nel contesto.

Nozioni di base

Hello Aurora

Gli esempi di codice seguenti mostrano come iniziare a utilizzare Aurora.

SDK per Java 2.x

### Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rds.RdsClient;
import software.amazon.awssdk.services.rds.paginators.DescribeDBClustersIterable;

public class DescribeDbClusters {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
```

```
RdsClient rdsClient = RdsClient.builder()
    .region(region)
    .build();

describeClusters(rdsClient);
rdsClient.close();
}

public static void describeClusters(RdsClient rdsClient) {
    DescribeDBClustersIterable clustersIterable =
rdsClient.describeDBClustersPaginator();
    clustersIterable.stream()
        .flatMap(r -> r.dbClusters().stream())
        .forEach(cluster -> System.out
            .println("Database name: " + cluster.databaseName() + " Arn
= " + cluster.dbClusterArn()));
}
}
```

- Per informazioni dettagliate sull'API, consulta [DescribeDBClusters](#) nella Documentazione di riferimento delle API di AWS SDK for Java 2.x .

## Argomenti

- [Azioni](#)
- [Scenari](#)

## Azioni

### CreateDBCluster

Il seguente esempio di codice mostra come usare `CreateDBCluster`.

SDK per Java 2.x

#### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public static String createDBCluster(RdsClient rdsClient, String
dbParameterGroupFamily, String dbName,
    String dbClusterIdentifier, String userName, String password) {
    try {
        CreateDbClusterRequest clusterRequest = CreateDbClusterRequest.builder()
            .databaseName(dbName)
            .dbClusterIdentifier(dbClusterIdentifier)
            .dbClusterParameterGroupName(dbParameterGroupFamily)
            .engine("aurora-mysql")
            .masterUsername(userName)
            .masterUserPassword(password)
            .build();

        CreateDbClusterResponse response =
rdsClient.createDBCluster(clusterRequest);
        return response.dbCluster().dbClusterArn();

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
    return "";
}
```

- Per informazioni dettagliate sull'API, consulta [CreateDBCluster](#) nella Documentazione di riferimento delle API di AWS SDK for Java 2.x .

## CreateDBClusterParameterGroup

Il seguente esempio di codice mostra come usare `CreateDBClusterParameterGroup`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public static void createDBClusterParameterGroup(RdsClient rdsClient, String
dbClusterGroupName,
    String dbParameterGroupFamily) {
    try {
        CreateDbClusterParameterGroupRequest groupRequest =
CreateDbClusterParameterGroupRequest.builder()
            .dbClusterParameterGroupName(dbClusterGroupName)
            .dbParameterGroupFamily(dbParameterGroupFamily)
            .description("Created by using the AWS SDK for Java")
            .build();

        CreateDbClusterParameterGroupResponse response =
rdsClient.createDBClusterParameterGroup(groupRequest);
        System.out.println("The group name is " +
response.dbClusterParameterGroup().dbClusterParameterGroupName());

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}
```

- Per i dettagli sull'API, consulta [CreateDB ClusterParameterGroup](#) in AWS SDK for Java 2.x API Reference.

## CreateDBClusterSnapshot

Il seguente esempio di codice mostra come utilizzare `CreateDBClusterSnapshot`

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public static void createDBClusterSnapshot(RdsClient rdsClient, String
dbInstanceClusterIdentifier,
    String dbSnapshotIdentifier) {
```

```
try {
    CreateDbClusterSnapshotRequest snapshotRequest =
CreateDbClusterSnapshotRequest.builder()
        .dbClusterIdentifier(dbInstanceClusterIdentifier)
        .dbClusterSnapshotIdentifier(dbSnapshotIdentifier)
        .build();

    CreateDbClusterSnapshotResponse response =
rdsClient.createDBClusterSnapshot(snapshotRequest);
    System.out.println("The Snapshot ARN is " +
response.dbClusterSnapshot().dbClusterSnapshotArn());

} catch (RdsException e) {
    System.out.println(e.getLocalizedMessage());
    System.exit(1);
}
}
```

- Per i dettagli sull'API, consulta [CreateDB ClusterSnapshot](#) in AWS SDK for Java 2.x API Reference.

## CreateDBInstance

Il seguente esempio di codice mostra come utilizzare `CreateDBInstance`

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public static String createDBInstanceCluster(RdsClient rdsClient,
    String dbInstanceIdentifier,
    String dbInstanceClusterIdentifier,
    String instanceClass) {
    try {
        CreateDbInstanceRequest instanceRequest =
CreateDbInstanceRequest.builder()
```

```

        .dbInstanceIdentifier(dbInstanceIdentifier)
        .dbClusterIdentifier(dbInstanceClusterIdentifier)
        .engine("aurora-mysql")
        .dbInstanceClass(instanceClass)
        .build();

        CreateDbInstanceResponse response =
rdsClient.createDBInstance(instanceRequest);
        System.out.print("The status is " +
response.dbInstance().dbInstanceStatus());
        return response.dbInstance().dbInstanceArn();

    } catch (RdsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return "";
}

```

- Per informazioni dettagliate sull'API, consulta [CreateDBInstance](#) nella Documentazione di riferimento delle API di AWS SDK for Java 2.x .

## DeleteDBCluster

Il seguente esempio di codice mostra come usare `DeleteDBCluster`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```

public static void deleteCluster(RdsClient rdsClient, String
dbInstanceClusterIdentifier) {
    try {
        DeleteDbClusterRequest deleteDbClusterRequest =
DeleteDbClusterRequest.builder()
            .dbClusterIdentifier(dbInstanceClusterIdentifier)

```



```
        .skipFinalSnapshot(true)
        .build();

        rdsClient.deleteDBCluster(deleteDbClusterRequest);
        System.out.println(dbInstanceClusterIdentifier + " was deleted!");
    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}
```

- Per informazioni dettagliate sull'API, consulta la sezione [DeleteDBCluster](#) nella Documentazione di riferimento delle API di AWS SDK for Java 2.x .

## DeleteDBClusterParameterGroup

Il seguente esempio di codice mostra come usare `DeleteDBClusterParameterGroup`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public static void deleteDBClusterGroup(RdsClient rdsClient, String
dbClusterGroupName, String clusterDBARN)
    throws InterruptedException {
    try {
        boolean isDataDel = false;
        boolean didFind;
        String instanceARN;

        // Make sure that the database has been deleted.
        while (!isDataDel) {
            DescribeDbInstancesResponse response =
rdsClient.describeDBInstances();
            List<DBInstance> instanceList = response.dbInstances();
```

```

        int listSize = instanceList.size();
        didFind = false;
        int index = 1;
        for (DBInstance instance : instanceList) {
            instanceARN = instance.dbInstanceArn();
            if (instanceARN.compareTo(clusterDBARN) == 0) {
                System.out.println(clusterDBARN + " still exists");
                didFind = true;
            }
            if ((index == listSize) && (!didFind)) {
                // Went through the entire list and did not find the
database ARN.
                isDataDel = true;
            }
            Thread.sleep(sleepTime * 1000);
            index++;
        }
    }

    DeleteDbClusterParameterGroupRequest clusterParameterGroupRequest =
DeleteDbClusterParameterGroupRequest
        .builder()
        .dbClusterParameterGroupName(dbClusterGroupName)
        .build();

    rdsClient.deleteDBClusterParameterGroup(clusterParameterGroupRequest);
    System.out.println(dbClusterGroupName + " was deleted.");

} catch (RdsException e) {
    System.out.println(e.getLocalizedMessage());
    System.exit(1);
}
}

```

- Per i dettagli sull'API, consulta [DeleteDB ClusterParameterGroup](#) in AWS SDK for Java 2.x API Reference.

## DeleteDBInstance

Il seguente esempio di codice mostra come utilizzare DeleteDBInstance

## SDK per Java 2.x

 Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public static void deleteDatabaseInstance(RdsClient rdsClient, String
dbInstanceIdentifier) {
    try {
        DeleteDbInstanceRequest deleteDbInstanceRequest =
DeleteDbInstanceRequest.builder()
            .dbInstanceIdentifier(dbInstanceIdentifier)
            .deleteAutomatedBackups(true)
            .skipFinalSnapshot(true)
            .build();

        DeleteDbInstanceResponse response =
rdsClient.deleteDBInstance(deleteDbInstanceRequest);
        System.out.println("The status of the database is " +
response.dbInstance().dbInstanceStatus());

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}
```

- Per informazioni dettagliate sull'API, consulta [DeleteDBInstance](#) nella Documentazione di riferimento delle API di AWS SDK for Java 2.x .

## DescribeDBClusterParameterGroups

Il seguente esempio di codice mostra come usare `DescribeDBClusterParameterGroups`.

## SDK per Java 2.x

 Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public static void describeDbClusterParameterGroups(RdsClient rdsClient, String
dbClusterGroupName) {
    try {
        DescribeDbClusterParameterGroupsRequest groupsRequest =
DescribeDbClusterParameterGroupsRequest.builder()
            .dbClusterParameterGroupName(dbClusterGroupName)
            .maxRecords(20)
            .build();

        List<DBClusterParameterGroup> groups =
rdsClient.describeDBClusterParameterGroups(groupsRequest)
            .dbClusterParameterGroups();
        for (DBClusterParameterGroup group : groups) {
            System.out.println("The group name is " +
group.dbClusterParameterGroupName());
            System.out.println("The group ARN is " +
group.dbClusterParameterGroupArn());
        }

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}
```

- Per i dettagli sull'API, consulta [DescribeDB ClusterParameterGroups](#) in AWS SDK for Java 2.x API Reference.

## DescribeDBClusterParameters

Il seguente esempio di codice mostra come utilizzare `DescribeDBClusterParameters`

## SDK per Java 2.x

 Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public static void describeDbClusterParameters(RdsClient rdsClient, String
dbClusterGroupName, int flag) {
    try {
        DescribeDbClusterParametersRequest dbParameterGroupsRequest;
        if (flag == 0) {
            dbParameterGroupsRequest =
DescribeDbClusterParametersRequest.builder()
                .dbClusterParameterGroupName(dbClusterGroupName)
                .build();
        } else {
            dbParameterGroupsRequest =
DescribeDbClusterParametersRequest.builder()
                .dbClusterParameterGroupName(dbClusterGroupName)
                .source("user")
                .build();
        }

        DescribeDbClusterParametersResponse response = rdsClient
            .describeDBClusterParameters(dbParameterGroupsRequest);
        List<Parameter> dbParameters = response.parameters();
        String paraName;
        for (Parameter para : dbParameters) {
            // Only print out information about either auto_increment_offset or
            // auto_increment_increment.
            paraName = para.parameterName();
            if ((paraName.compareTo("auto_increment_offset") == 0)
                || (paraName.compareTo("auto_increment_increment ") == 0)) {
                System.out.println("*** The parameter name is " + paraName);
                System.out.println("*** The parameter value is " +
para.parameterValue());
                System.out.println("*** The parameter data type is " +
para.dataType());
                System.out.println("*** The parameter description is " +
para.description());
            }
        }
    }
}
```

```
        System.out.println("*** The parameter allowed values is " +
para.allowedValues());
    }
}

} catch (RdsException e) {
    System.out.println(e.getLocalizedMessage());
    System.exit(1);
}
}
```

- Per i dettagli sull'API, consulta [DescribeDB ClusterParameters](#) in AWS SDK for Java 2.x API Reference.

## DescribeDBClusterSnapshots

Il seguente esempio di codice mostra come utilizzare `DescribeDBClusterSnapshots`

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public static void waitForSnapshotReady(RdsClient rdsClient, String
dbSnapshotIdentifier,
    String dbInstanceClusterIdentifier) {
    try {
        boolean snapshotReady = false;
        String snapshotReadyStr;
        System.out.println("Waiting for the snapshot to become available.");

        DescribeDbClusterSnapshotsRequest snapshotsRequest =
DescribeDbClusterSnapshotsRequest.builder()
            .dbClusterSnapshotIdentifier(dbSnapshotIdentifier)
            .dbClusterIdentifier(dbInstanceClusterIdentifier)
            .build();
```

```
        while (!snapshotReady) {
            DescribeDbClusterSnapshotsResponse response =
rdsClient.describeDBClusterSnapshots(snapshotRequest);
            List<DBClusterSnapshot> snapshotList =
response.dbClusterSnapshots();
            for (DBClusterSnapshot snapshot : snapshotList) {
                snapshotReadyStr = snapshot.status();
                if (snapshotReadyStr.contains("available")) {
                    snapshotReady = true;
                } else {
                    System.out.println(".");
                    Thread.sleep(sleepTime * 5000);
                }
            }
        }

        System.out.println("The Snapshot is available!");

    } catch (RdsException | InterruptedException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}
```

- Per i dettagli sull'API, consulta [DescribeDB ClusterSnapshots](#) in AWS SDK for Java 2.x API Reference.

## DescribeDBClusters

Il seguente esempio di codice mostra come utilizzare `DescribeDBClusters`

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public static void describeDbClusterParameters(RdsClient rdsClient, String
dbClusterGroupName, int flag) {
    try {
        DescribeDbClusterParametersRequest dbParameterGroupsRequest;
        if (flag == 0) {
            dbParameterGroupsRequest =
DescribeDbClusterParametersRequest.builder()
                .dbClusterParameterGroupName(dbClusterGroupName)
                .build();
        } else {
            dbParameterGroupsRequest =
DescribeDbClusterParametersRequest.builder()
                .dbClusterParameterGroupName(dbClusterGroupName)
                .source("user")
                .build();
        }

        DescribeDbClusterParametersResponse response = rdsClient
            .describeDBClusterParameters(dbParameterGroupsRequest);
        List<Parameter> dbParameters = response.parameters();
        String paraName;
        for (Parameter para : dbParameters) {
            // Only print out information about either auto_increment_offset or
            // auto_increment_increment.
            paraName = para.parameterName();
            if ((paraName.compareTo("auto_increment_offset") == 0)
                || (paraName.compareTo("auto_increment_increment ") == 0)) {
                System.out.println("*** The parameter name is " + paraName);
                System.out.println("*** The parameter value is " +
para.parameterValue());
                System.out.println("*** The parameter data type is " +
para.dataType());
                System.out.println("*** The parameter description is " +
para.description());
                System.out.println("*** The parameter allowed values is " +
para.allowedValues());
            }
        }

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}
```



```
}
```

- Per informazioni dettagliate sull'API, consulta [DescribeDBClusters](#) nella Documentazione di riferimento delle API di AWS SDK for Java 2.x .

## DescribeDBEngineVersions

Il seguente esempio di codice mostra come usare `DescribeDBEngineVersions`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public static void describeDBEngines(RdsClient rdsClient) {
    try {
        DescribeDbEngineVersionsRequest engineVersionsRequest =
DescribeDbEngineVersionsRequest.builder()
            .engine("aurora-mysql")
            .defaultOnly(true)
            .maxRecords(20)
            .build();

        DescribeDbEngineVersionsResponse response =
rdsClient.describeDBEngineVersions(engineVersionsRequest);
        List<DBEngineVersion> engines = response.dbEngineVersions();

        // Get all DBEngineVersion objects.
        for (DBEngineVersion engineOb : engines) {
            System.out.println("The name of the DB parameter group family for
the database engine is "
                + engineOb.dbParameterGroupFamily());
            System.out.println("The name of the database engine " +
engineOb.engine());
            System.out.println("The version number of the database engine " +
engineOb.engineVersion());
        }
    }
}
```

```
    } catch (RdsException e) {  
        System.out.println(e.getLocalizedMessage());  
        System.exit(1);  
    }  
}
```

- Per i dettagli sull'API, consulta [DescribeDB EngineVersions](#) in AWS SDK for Java 2.x API Reference.

## DescribeDBInstances

Il seguente esempio di codice mostra come utilizzare `DescribeDBInstances`

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
// Waits until the database instance is available.  
public static void waitForInstanceReady(RdsClient rdsClient, String  
dbClusterIdentifier) {  
    boolean instanceReady = false;  
    String instanceReadyStr;  
    System.out.println("Waiting for instance to become available.");  
    try {  
        DescribeDbClustersRequest instanceRequest =  
DescribeDbClustersRequest.builder()  
            .dbClusterIdentifier(dbClusterIdentifier)  
            .build();  
  
        while (!instanceReady) {  
            DescribeDbClustersResponse response =  
rdsClient.describeDBClusters(instanceRequest);  
            List<DBCluster> clusterList = response.dbClusters();  
            for (DBCluster cluster : clusterList) {  
                instanceReadyStr = cluster.status();  
            }  
        }  
    }  
}
```

```
        if (instanceReadyStr.contains("available")) {
            instanceReady = true;
        } else {
            System.out.print(".");
            Thread.sleep(sleepTime * 1000);
        }
    }
}
System.out.println("Database cluster is available!");

} catch (RdsException | InterruptedException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
}
```

- Per informazioni dettagliate sull'API, consulta [DescribeDBInstances](#) nella Documentazione di riferimento delle API di AWS SDK for Java 2.x .

## DescribeOrderableDBInstanceOptions

Il seguente esempio di codice mostra come usare `DescribeOrderableDBInstanceOptions`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public static void describeDBEngines(RdsClient rdsClient) {
    try {
        DescribeDbEngineVersionsRequest engineVersionsRequest =
DescribeDbEngineVersionsRequest.builder()
            .engine("aurora-mysql")
            .defaultOnly(true)
            .maxRecords(20)
            .build();
```

```
DescribeDbEngineVersionsResponse response =
rdsClient.describeDBEngineVersions(engineVersionsRequest);
List<DBEngineVersion> engines = response.dbEngineVersions();

// Get all DBEngineVersion objects.
for (DBEngineVersion engineOb : engines) {
    System.out.println("The name of the DB parameter group family for
the database engine is "
        + engineOb.dbParameterGroupFamily());
    System.out.println("The name of the database engine " +
engineOb.engine());
    System.out.println("The version number of the database engine " +
engineOb.engineVersion());
}

} catch (RdsException e) {
    System.out.println(e.getLocalizedMessage());
    System.exit(1);
}
}
```

- Per i dettagli sull'API, consulta [DescribeOrderableDB InstanceOptions](#) in AWS SDK for Java 2.x API Reference.

## ModifyDBClusterParameterGroup

Il seguente esempio di codice mostra come utilizzare `ModifyDBClusterParameterGroup`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public static void describeDbClusterParameterGroups(RdsClient rdsClient, String
dbClusterGroupName) {
    try {
```

```
DescribeDbClusterParameterGroupsRequest groupsRequest =
DescribeDbClusterParameterGroupsRequest.builder()
    .dbClusterParameterGroupName(dbClusterGroupName)
    .maxRecords(20)
    .build();

List<DBClusterParameterGroup> groups =
rdsClient.describeDBClusterParameterGroups(groupsRequest)
    .dbClusterParameterGroups();
for (DBClusterParameterGroup group : groups) {
    System.out.println("The group name is " +
group.dbClusterParameterGroupName());
    System.out.println("The group ARN is " +
group.dbClusterParameterGroupArn());
}

} catch (RdsException e) {
    System.out.println(e.getLocalizedMessage());
    System.exit(1);
}
}
```

- Per i dettagli sull'API, consulta [ModifyDB ClusterParameterGroup](#) in AWS SDK for Java 2.x API Reference.

## Scenari

### Uso dei cluster di database

L'esempio di codice seguente mostra come:

- Crea un gruppo di parametri del cluster di database Aurora personalizzati e imposta i relativi valori.
- Crea un cluster di database che utilizza il gruppo di parametri.
- Crea un'istanza database che contiene un database.
- Acquisisci uno snapshot del cluster di database, quindi elimina le risorse.

## SDK per Java 2.x

 Note

C'è di più su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
/**
 * Before running this Java (v2) code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * This example requires an AWS Secrets Manager secret that contains the
 * database credentials. If you do not create a
 * secret, this example will not work. For details, see:
 *
 * https://docs.aws.amazon.com/secretsmanager/latest/userguide/integrating_how-
 * services-use-secrets_RS.html
 *
 * This Java example performs the following tasks:
 *
 * 1. Gets available engine families for Amazon Aurora MySQL-Compatible Edition
 * by calling the DescribeDbEngineVersions(Engine='aurora-mysql') method.
 * 2. Selects an engine family and creates a custom DB cluster parameter group
 * by invoking the describeDBClusterParameters method.
 * 3. Gets the parameter groups by invoking the describeDBClusterParameterGroups
 * method.
 * 4. Gets parameters in the group by invoking the describeDBClusterParameters
 * method.
 * 5. Modifies the auto_increment_offset parameter by invoking the
 * modifyDbClusterParameterGroupRequest method.
 * 6. Gets and displays the updated parameters.
 * 7. Gets a list of allowed engine versions by invoking the
 * describeDbEngineVersions method.
 * 8. Creates an Aurora DB cluster database cluster that contains a MySQL
 * database.
 * 9. Waits for DB instance to be ready.
 * 10. Gets a list of instance classes available for the selected engine.
```

```

* 11. Creates a database instance in the cluster.
* 12. Waits for DB instance to be ready.
* 13. Creates a snapshot.
* 14. Waits for DB snapshot to be ready.
* 15. Deletes the DB cluster.
* 16. Deletes the DB cluster group.
*/
public class AuroraScenario {
    public static long sleepTime = 20;
    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    public static void main(String[] args) throws InterruptedException {
        final String usage = "\n" +
            "Usage:\n" +
            "    <dbClusterGroupName> <dbParameterGroupFamily>
<dbInstanceClusterIdentifier> <dbInstanceIdentifier> <dbName>
<dbSnapshotIdentifier><secretName>"
            +
            "Where:\n" +
            "    dbClusterGroupName - The name of the DB cluster parameter
group. \n" +
            "    dbParameterGroupFamily - The DB cluster parameter group family
name (for example, aurora-mysql5.7). \n"
            +
            "    dbInstanceClusterIdentifier - The instance cluster identifier
value.\n" +
            "    dbInstanceIdentifier - The database instance identifier.\n" +
            "    dbName - The database name.\n" +
            "    dbSnapshotIdentifier - The snapshot identifier.\n" +
            "    secretName - The name of the AWS Secrets Manager secret that
contains the database credentials\`\n";
        ;

        if (args.length != 7) {
            System.out.println(usage);
            System.exit(1);
        }

        String dbClusterGroupName = args[0];
        String dbParameterGroupFamily = args[1];
        String dbInstanceClusterIdentifier = args[2];
        String dbInstanceIdentifier = args[3];
        String dbName = args[4];
        String dbSnapshotIdentifier = args[5];

```

```
String secretName = args[6];

// Retrieve the database credentials using AWS Secrets Manager.
Gson gson = new Gson();
User user = gson.fromJson(String.valueOf(getSecretValues(secretName)),
User.class);
String username = user.getUsername();
String userPassword = user.getPassword();

Region region = Region.US_WEST_2;
RdsClient rdsClient = RdsClient.builder()
    .region(region)
    .build();

System.out.println(DASHES);
System.out.println("Welcome to the Amazon Aurora example scenario.");
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("1. Return a list of the available DB engines");
describeDBEngines(rdsClient);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("2. Create a custom parameter group");
createDBClusterParameterGroup(rdsClient, dbClusterGroupName,
dbParameterGroupFamily);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("3. Get the parameter group");
describeDbClusterParameterGroups(rdsClient, dbClusterGroupName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("4. Get the parameters in the group");
describeDbClusterParameters(rdsClient, dbClusterGroupName, 0);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("5. Modify the auto_increment_offset parameter");
modifyDBClusterParas(rdsClient, dbClusterGroupName);
System.out.println(DASHES);
```



```
System.out.println(DASHES);
System.out.println("6. Display the updated parameter value");
describeDbClusterParameters(rdsClient, dbClusterGroupName, -1);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("7. Get a list of allowed engine versions");
getAllowedEngines(rdsClient, dbParameterGroupFamily);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("8. Create an Aurora DB cluster database");
String arnClusterVal = createDBCluster(rdsClient, dbClusterGroupName,
dbName, dbInstanceClusterIdentifier,
    username, userPassword);
System.out.println("The ARN of the cluster is " + arnClusterVal);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("9. Wait for DB instance to be ready");
waitForInstanceReady(rdsClient, dbInstanceClusterIdentifier);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("10. Get a list of instance classes available for the
selected engine");
String instanceClass = getListInstanceClasses(rdsClient);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("11. Create a database instance in the cluster.");
String clusterDBARN = createDBInstanceCluster(rdsClient,
dbInstanceIdentifier, dbInstanceClusterIdentifier,
    instanceClass);
System.out.println("The ARN of the database is " + clusterDBARN);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("12. Wait for DB instance to be ready");
waitDBInstanceReady(rdsClient, dbInstanceIdentifier);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("13. Create a snapshot");
```

```
        createDBClusterSnapshot(rdsClient, dbInstanceClusterIdentifier,
dbSnapshotIdentifier);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("14. Wait for DB snapshot to be ready");
        waitForSnapshotReady(rdsClient, dbSnapshotIdentifier,
dbInstanceClusterIdentifier);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("14. Delete the DB instance");
        deleteDatabaseInstance(rdsClient, dbInstanceIdentifier);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("15. Delete the DB cluster");
        deleteCluster(rdsClient, dbInstanceClusterIdentifier);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("16. Delete the DB cluster group");
        deleteDBClusterGroup(rdsClient, dbClusterGroupName, clusterDBARN);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("The Scenario has successfully completed.");
        System.out.println(DASHES);
        rdsClient.close();
    }

    private static SecretsManagerClient getSecretClient() {
        Region region = Region.US_WEST_2;
        return SecretsManagerClient.builder()
            .region(region)

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

    private static String getSecretValues(String secretName) {
        SecretsManagerClient secretClient = getSecretClient();
        GetSecretValueRequest valueRequest = GetSecretValueRequest.builder()
            .secretId(secretName)
```

```

        .build();

        GetSecretValueResponse valueResponse =
secretClient.getSecretValue(valueRequest);
        return valueResponse.secretString();
    }

    public static void deleteDBClusterGroup(RdsClient rdsClient, String
dbClusterGroupName, String clusterDBARN)
        throws InterruptedException {
    try {
        boolean isDataDel = false;
        boolean didFind;
        String instanceARN;

        // Make sure that the database has been deleted.
        while (!isDataDel) {
            DescribeDbInstancesResponse response =
rdsClient.describeDBInstances();
            List<DBInstance> instanceList = response.dbInstances();
            int listSize = instanceList.size();
            didFind = false;
            int index = 1;
            for (DBInstance instance : instanceList) {
                instanceARN = instance.dbInstanceArn();
                if (instanceARN.compareTo(clusterDBARN) == 0) {
                    System.out.println(clusterDBARN + " still exists");
                    didFind = true;
                }
            }
            if ((index == listSize) && (!didFind)) {
                // Went through the entire list and did not find the
database ARN.

                isDataDel = true;
            }
            Thread.sleep(sleepTime * 1000);
            index++;
        }
    }

        DeleteDbClusterParameterGroupRequest clusterParameterGroupRequest =
DeleteDbClusterParameterGroupRequest
            .builder()
            .dbClusterParameterGroupName(dbClusterGroupName)
            .build();

```

```
        rdsClient.deleteDBClusterParameterGroup(clusterParameterGroupRequest);
        System.out.println(dbClusterGroupName + " was deleted.");

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}

public static void deleteCluster(RdsClient rdsClient, String
dbInstanceClusterIdentifier) {
    try {
        DeleteDbClusterRequest deleteDbClusterRequest =
DeleteDbClusterRequest.builder()
            .dbClusterIdentifier(dbInstanceClusterIdentifier)
            .skipFinalSnapshot(true)
            .build();

        rdsClient.deleteDBCluster(deleteDbClusterRequest);
        System.out.println(dbInstanceClusterIdentifier + " was deleted!");

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}

public static void deleteDatabaseInstance(RdsClient rdsClient, String
dbInstanceIdentifier) {
    try {
        DeleteDbInstanceRequest deleteDbInstanceRequest =
DeleteDbInstanceRequest.builder()
            .dbInstanceIdentifier(dbInstanceIdentifier)
            .deleteAutomatedBackups(true)
            .skipFinalSnapshot(true)
            .build();

        DeleteDbInstanceResponse response =
rdsClient.deleteDBInstance(deleteDbInstanceRequest);
        System.out.println("The status of the database is " +
response.dbInstance().dbInstanceStatus());

    } catch (RdsException e) {
```

```
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}

public static void waitForSnapshotReady(RdsClient rdsClient, String
dbSnapshotIdentifier,
    String dbInstanceClusterIdentifier) {
    try {
        boolean snapshotReady = false;
        String snapshotReadyStr;
        System.out.println("Waiting for the snapshot to become available.");

        DescribeDbClusterSnapshotsRequest snapshotsRequest =
DescribeDbClusterSnapshotsRequest.builder()
            .dbClusterSnapshotIdentifier(dbSnapshotIdentifier)
            .dbClusterIdentifier(dbInstanceClusterIdentifier)
            .build();

        while (!snapshotReady) {
            DescribeDbClusterSnapshotsResponse response =
rdsClient.describeDBClusterSnapshots(snapshotsRequest);
            List<DBClusterSnapshot> snapshotList =
response.dbClusterSnapshots();
            for (DBClusterSnapshot snapshot : snapshotList) {
                snapshotReadyStr = snapshot.status();
                if (snapshotReadyStr.contains("available")) {
                    snapshotReady = true;
                } else {
                    System.out.println(".");
                    Thread.sleep(sleepTime * 5000);
                }
            }
        }

        System.out.println("The Snapshot is available!");

    } catch (RdsException | InterruptedException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}
```

```
public static void createDBClusterSnapshot(RdsClient rdsClient, String
dbInstanceClusterIdentifier,
    String dbSnapshotIdentifier) {
    try {
        CreateDbClusterSnapshotRequest snapshotRequest =
CreateDbClusterSnapshotRequest.builder()
            .dbClusterIdentifier(dbInstanceClusterIdentifier)
            .dbClusterSnapshotIdentifier(dbSnapshotIdentifier)
            .build();

        CreateDbClusterSnapshotResponse response =
rdsClient.createDBClusterSnapshot(snapshotRequest);
        System.out.println("The Snapshot ARN is " +
response.dbClusterSnapshot().dbClusterSnapshotArn());

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}

public static void waitDBInstanceReady(RdsClient rdsClient, String
dbInstanceIdentifier) {
    boolean instanceReady = false;
    String instanceReadyStr;
    System.out.println("Waiting for instance to become available.");
    try {
        DescribeDbInstancesRequest instanceRequest =
DescribeDbInstancesRequest.builder()
            .dbInstanceIdentifier(dbInstanceIdentifier)
            .build();

        String endpoint = "";
        while (!instanceReady) {
            DescribeDbInstancesResponse response =
rdsClient.describeDBInstances(instanceRequest);
            List<DBInstance> instanceList = response.dbInstances();
            for (DBInstance instance : instanceList) {
                instanceReadyStr = instance.dbInstanceStatus();
                if (instanceReadyStr.contains("available")) {
                    endpoint = instance.endpoint().address();
                    instanceReady = true;
                } else {
                    System.out.print(".");
                }
            }
        }
    }
}
```

```

        Thread.sleep(sleepTime * 1000);
    }
}
}
System.out.println("Database instance is available! The connection
endpoint is " + endpoint);

} catch (RdsException | InterruptedException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
}

public static String createDBInstanceCluster(RdsClient rdsClient,
    String dbInstanceIdentifier,
    String dbInstanceClusterIdentifier,
    String instanceClass) {
    try {
        CreateDbInstanceRequest instanceRequest =
CreateDbInstanceRequest.builder()
        .dbInstanceIdentifier(dbInstanceIdentifier)
        .dbClusterIdentifier(dbInstanceClusterIdentifier)
        .engine("aurora-mysql")
        .dbInstanceClass(instanceClass)
        .build();

        CreateDbInstanceResponse response =
rdsClient.createDBInstance(instanceRequest);
        System.out.print("The status is " +
response.dbInstance().dbInstanceStatus());
        return response.dbInstance().dbInstanceArn();

    } catch (RdsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return "";
}

public static String getListInstanceClasses(RdsClient rdsClient) {
    try {
        DescribeOrderableDbInstanceOptionsRequest optionsRequest =
DescribeOrderableDbInstanceOptionsRequest
        .builder()

```

```

        .engine("aurora-mysql")
        .maxRecords(20)
        .build();

DescribeOrderableDbInstanceOptionsResponse response = rdsClient
    .describeOrderableDBInstanceOptions(optionsRequest);
List<OrderableDBInstanceOption> instanceOptions =
response.orderableDBInstanceOptions();
String instanceClass = "";
for (OrderableDBInstanceOption instanceOption : instanceOptions) {
    instanceClass = instanceOption.dbInstanceClass();
    System.out.println("The instance class is " +
instanceOption.dbInstanceClass());
    System.out.println("The engine version is " +
instanceOption.engineVersion());
}
return instanceClass;

} catch (RdsException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
return "";
}

// Waits until the database instance is available.
public static void waitForInstanceReady(RdsClient rdsClient, String
dbClusterIdentifier) {
    boolean instanceReady = false;
    String instanceReadyStr;
    System.out.println("Waiting for instance to become available.");
    try {
        DescribeDbClustersRequest instanceRequest =
DescribeDbClustersRequest.builder()
            .dbClusterIdentifier(dbClusterIdentifier)
            .build();

        while (!instanceReady) {
            DescribeDbClustersResponse response =
rdsClient.describeDBClusters(instanceRequest);
            List<DBCluster> clusterList = response.dbClusters();
            for (DBCluster cluster : clusterList) {
                instanceReadyStr = cluster.status();
                if (instanceReadyStr.contains("available")) {

```



```

        instanceReady = true;
    } else {
        System.out.print(".");
        Thread.sleep(sleepTime * 1000);
    }
}
}
System.out.println("Database cluster is available!");

} catch (RdsException | InterruptedException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
}

public static String createDBCluster(RdsClient rdsClient, String
dbParameterGroupFamily, String dbName,
String dbClusterIdentifier, String userName, String password) {
    try {
        CreateDbClusterRequest clusterRequest = CreateDbClusterRequest.builder()
            .databaseName(dbName)
            .dbClusterIdentifier(dbClusterIdentifier)
            .dbClusterParameterGroupName(dbParameterGroupFamily)
            .engine("aurora-mysql")
            .masterUsername(userName)
            .masterUserPassword(password)
            .build();

        CreateDbClusterResponse response =
rdsClient.createDBCluster(clusterRequest);
        return response.dbCluster().dbClusterArn();

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
    return "";
}

// Get a list of allowed engine versions.
public static void getAllowedEngines(RdsClient rdsClient, String
dbParameterGroupFamily) {
    try {

```

```

        DescribeDbEngineVersionsRequest versionsRequest =
DescribeDbEngineVersionsRequest.builder()
    .dbParameterGroupFamily(dbParameterGroupFamily)
    .engine("aurora-mysql")
    .build();

        DescribeDbEngineVersionsResponse response =
rdsClient.describeDBEngineVersions(versionsRequest);
        List<DBEngineVersion> dbEngines = response.dbEngineVersions();
        for (DBEngineVersion dbEngine : dbEngines) {
            System.out.println("The engine version is " +
dbEngine.engineVersion());
            System.out.println("The engine description is " +
dbEngine.dbEngineDescription());
        }

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}

// Modify the auto_increment_offset parameter.
public static void modifyDBClusterParas(RdsClient rdsClient, String
dClusterGroupName) {
    try {
        Parameter parameter1 = Parameter.builder()
            .parameterName("auto_increment_offset")
            .applyMethod("immediate")
            .parameterValue("5")
            .build();

        List<Parameter> paraList = new ArrayList<>();
        paraList.add(parameter1);
        ModifyDbClusterParameterGroupRequest groupRequest =
ModifyDbClusterParameterGroupRequest.builder()
            .dbClusterParameterGroupName(dClusterGroupName)
            .parameters(paraList)
            .build();

        ModifyDbClusterParameterGroupResponse response =
rdsClient.modifyDBClusterParameterGroup(groupRequest);
        System.out.println(

```

```

        "The parameter group " + response.dbClusterParameterGroupName()
+ " was successfully modified");

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}

public static void describeDbClusterParameters(RdsClient rdsClient, String
dbClusterGroupName, int flag) {
    try {
        DescribeDbClusterParametersRequest dbParameterGroupsRequest;
        if (flag == 0) {
            dbParameterGroupsRequest =
DescribeDbClusterParametersRequest.builder()
                .dbClusterParameterGroupName(dbClusterGroupName)
                .build();
        } else {
            dbParameterGroupsRequest =
DescribeDbClusterParametersRequest.builder()
                .dbClusterParameterGroupName(dbClusterGroupName)
                .source("user")
                .build();
        }

        DescribeDbClusterParametersResponse response = rdsClient
            .describeDBClusterParameters(dbParameterGroupsRequest);
        List<Parameter> dbParameters = response.parameters();
        String paraName;
        for (Parameter para : dbParameters) {
            // Only print out information about either auto_increment_offset or
            // auto_increment_increment.
            paraName = para.parameterName();
            if ((paraName.compareTo("auto_increment_offset") == 0)
                || (paraName.compareTo("auto_increment_increment ") == 0)) {
                System.out.println("*** The parameter name is " + paraName);
                System.out.println("*** The parameter value is " +
para.parameterValue());
                System.out.println("*** The parameter data type is " +
para.dataType());
                System.out.println("*** The parameter description is " +
para.description());
            }
        }
    }
}

```

```

        System.out.println("*** The parameter allowed values is " +
para.allowedValues());
    }
}

} catch (RdsException e) {
    System.out.println(e.getLocalizedMessage());
    System.exit(1);
}
}

public static void describeDbClusterParameterGroups(RdsClient rdsClient, String
dbClusterGroupName) {
    try {
        DescribeDbClusterParameterGroupsRequest groupsRequest =
DescribeDbClusterParameterGroupsRequest.builder()
            .dbClusterParameterGroupName(dbClusterGroupName)
            .maxRecords(20)
            .build();

        List<DBClusterParameterGroup> groups =
rdsClient.describeDBClusterParameterGroups(groupsRequest)
            .dbClusterParameterGroups();
        for (DBClusterParameterGroup group : groups) {
            System.out.println("The group name is " +
group.dbClusterParameterGroupName());
            System.out.println("The group ARN is " +
group.dbClusterParameterGroupArn());
        }

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}

public static void createDBClusterParameterGroup(RdsClient rdsClient, String
dbClusterGroupName,
    String dbParameterGroupFamily) {
    try {
        CreateDbClusterParameterGroupRequest groupRequest =
CreateDbClusterParameterGroupRequest.builder()
            .dbClusterParameterGroupName(dbClusterGroupName)
            .dbParameterGroupFamily(dbParameterGroupFamily)

```

```
        .description("Created by using the AWS SDK for Java")
        .build();

        CreateDbClusterParameterGroupResponse response =
rdsClient.createDBClusterParameterGroup(groupRequest);
        System.out.println("The group name is " +
response.dbClusterParameterGroup().dbClusterParameterGroupName());

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}

public static void describeDBEngines(RdsClient rdsClient) {
    try {
        DescribeDbEngineVersionsRequest engineVersionsRequest =
DescribeDbEngineVersionsRequest.builder()
            .engine("aurora-mysql")
            .defaultOnly(true)
            .maxRecords(20)
            .build();

        DescribeDbEngineVersionsResponse response =
rdsClient.describeDBEngineVersions(engineVersionsRequest);
        List<DBEngineVersion> engines = response.dbEngineVersions();

        // Get all DBEngineVersion objects.
        for (DBEngineVersion engineOb : engines) {
            System.out.println("The name of the DB parameter group family for
the database engine is "
                + engineOb.dbParameterGroupFamily());
            System.out.println("The name of the database engine " +
engineOb.engine());
            System.out.println("The version number of the database engine " +
engineOb.engineVersion());
        }

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}
}
```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella Documentazione di riferimento delle API AWS SDK for Java 2.x .
  - [CreateDBCluster](#)
  - [Creato B ClusterParameterGroup](#)
  - [Creato DB ClusterSnapshot](#)
  - [CreateDBInstance](#)
  - [DeleteDBCluster](#)
  - [Eliminare DB ClusterParameterGroup](#)
  - [DeleteDBInstance](#)
  - [Descritto B ClusterParameterGroups](#)
  - [Descritto B ClusterParameters](#)
  - [Descritto B ClusterSnapshots](#)
  - [DescribeDBClusters](#)
  - [Descritto B EngineVersions](#)
  - [DescribeDBInstances](#)
  - [DescribeOrderableDB InstanceOptions](#)
  - [Modifica DB ClusterParameterGroup](#)

## Esempi di Auto Scaling con SDK for Java 2.x

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando AWS SDK for Java 2.x con Auto Scaling.

Le operazioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le operazioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati e negli esempi tra servizi.

Scenari: esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio.

Ogni esempio include un collegamento a GitHub, dove è possibile trovare istruzioni su come configurare ed eseguire il codice nel contesto.

## Nozioni di base

### Ciao Auto Scaling

I seguenti esempi di codice mostrano come iniziare a usare Auto Scaling.

### SDK per Java 2.x

#### Note

C'è altro su [GitHub](#) Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.autoscaling.AutoScalingClient;
import software.amazon.awssdk.services.autoscaling.model.AutoScalingGroup;
import
    software.amazon.awssdk.services.autoscaling.model.DescribeAutoScalingGroupsResponse;
import java.util.List;

/**
 * Before running this SDK for Java (v2) code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DescribeAutoScalingGroups {
    public static void main(String[] args) throws InterruptedException {
        AutoScalingClient autoScalingClient = AutoScalingClient.builder()
            .region(Region.US_EAST_1)
            .build();

        describeGroups(autoScalingClient);
    }

    public static void describeGroups(AutoScalingClient autoScalingClient) {
        DescribeAutoScalingGroupsResponse response =
autoScalingClient.describeAutoScalingGroups();
        List<AutoScalingGroup> groups = response.autoScalingGroups();
        groups.forEach(group -> {
```

```
        System.out.println("Group Name: " + group.autoScalingGroupName());
        System.out.println("Group ARN: " + group.autoScalingGroupARN());
    });
}
}
```

- Per i dettagli sull'API, consulta la [DescribeAutoScalingGroups](#) sezione AWS SDK for Java 2.x API Reference.

## Argomenti

- [Azioni](#)
- [Scenari](#)

## Azioni

### CreateAutoScalingGroup

Il seguente esempio di codice mostra come utilizzare `CreateAutoScalingGroup`.

#### SDK per Java 2.x

#### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.autoscaling.AutoScalingClient;
import software.amazon.awssdk.services.autoscaling.model.AutoScalingException;
import
    software.amazon.awssdk.services.autoscaling.model.CreateAutoScalingGroupRequest;
import
    software.amazon.awssdk.services.autoscaling.model.DescribeAutoScalingGroupsRequest;
import
    software.amazon.awssdk.services.autoscaling.model.DescribeAutoScalingGroupsResponse;
import
    software.amazon.awssdk.services.autoscaling.model.LaunchTemplateSpecification;
```



```
import software.amazon.awssdk.services.autoscaling.waiters.AutoScalingWaiter;

/**
 * Before running this SDK for Java (v2) code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateAutoScalingGroup {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                    <groupName> <launchTemplateName> <serviceLinkedRoleARN>
<vpcZoneId>

                Where:
                    groupName - The name of the Auto Scaling group.
                    launchTemplateName - The name of the launch template.\s
                    vpcZoneId - A subnet Id for a virtual private cloud (VPC) where
instances in the Auto Scaling group can be created.
                ""

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String groupName = args[0];
        String launchTemplateName = args[1];
        String vpcZoneId = args[2];
        AutoScalingClient autoScalingClient = AutoScalingClient.builder()
            .region(Region.US_EAST_1)
            .build();

        createAutoScalingGroup(autoScalingClient, groupName, launchTemplateName,
vpcZoneId);
        autoScalingClient.close();
    }

    public static void createAutoScalingGroup(AutoScalingClient autoScalingClient,
        String groupName,
```

```
        String launchTemplateName,
        String vpcZoneId) {

    try {
        AutoScalingWaiter waiter = autoScalingClient.waiter();
        LaunchTemplateSpecification templateSpecification =
LaunchTemplateSpecification.builder()
            .launchTemplateName(launchTemplateName)
            .build();

        CreateAutoScalingGroupRequest request =
CreateAutoScalingGroupRequest.builder()
            .autoScalingGroupName(groupName)
            .availabilityZones("us-east-1a")
            .launchTemplate(templateSpecification)
            .maxSize(1)
            .minSize(1)
            .vpcZoneIdentifier(vpcZoneId)
            .build();

        autoScalingClient.createAutoScalingGroup(request);
        DescribeAutoScalingGroupsRequest groupsRequest =
DescribeAutoScalingGroupsRequest.builder()
            .autoScalingGroupNames(groupName)
            .build();

        WaiterResponse<DescribeAutoScalingGroupsResponse> waiterResponse =
waiter
            .waitUntilGroupExists(groupsRequest);
        waiterResponse.matched().response().ifPresent(System.out::println);
        System.out.println("Auto Scaling Group created");

    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Per i dettagli sull'API, consulta la [CreateAutoScalingGroup](#) sezione AWS SDK for Java 2.x API Reference.

## DeleteAutoScalingGroup

Il seguente esempio di codice mostra come utilizzare `DeleteAutoScalingGroup`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.autoscaling.AutoScalingClient;
import software.amazon.awssdk.services.autoscaling.model.AutoScalingException;
import
    software.amazon.awssdk.services.autoscaling.model.DeleteAutoScalingGroupRequest;

/**
 * Before running this SDK for Java (v2) code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteAutoScalingGroup {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <groupName>

            Where:
                groupName - The name of the Auto Scaling group.
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String groupName = args[0];
```

```
        AutoScalingClient autoScalingClient = AutoScalingClient.builder()
            .region(Region.US_EAST_1)
            .build();

        deleteAutoScalingGroup(autoScalingClient, groupName);
        autoScalingClient.close();
    }

    public static void deleteAutoScalingGroup(AutoScalingClient autoScalingClient,
String groupName) {
        try {
            DeleteAutoScalingGroupRequest deleteAutoScalingGroupRequest =
DeleteAutoScalingGroupRequest.builder()
                .autoScalingGroupName(groupName)
                .forceDelete(true)
                .build();

            autoScalingClient.deleteAutoScalingGroup(deleteAutoScalingGroupRequest);
            System.out.println("You successfully deleted " + groupName);

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Per i dettagli sull'API, consulta la [DeleteAutoScalingGroup](#) sezione AWS SDK for Java 2.x API Reference.

## DescribeAutoScalingGroups

Il seguente esempio di codice mostra come utilizzare `DescribeAutoScalingGroups`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.autoscaling.AutoScalingClient;
import software.amazon.awssdk.services.autoscaling.model.AutoScalingException;
import software.amazon.awssdk.services.autoscaling.model.AutoScalingGroup;
import
    software.amazon.awssdk.services.autoscaling.model.DescribeAutoScalingGroupsResponse;
import
    software.amazon.awssdk.services.autoscaling.model.DescribeAutoScalingGroupsRequest;
import software.amazon.awssdk.services.autoscaling.model.Instance;
import java.util.List;

/**
 * Before running this SDK for Java (v2) code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DescribeAutoScalingInstances {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <groupName>

                Where:
                groupName - The name of the Auto Scaling group.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String groupName = args[0];
        AutoScalingClient autoScalingClient = AutoScalingClient.builder()
            .region(Region.US_EAST_1)
            .build();

        String instanceId = getAutoScaling(autoScalingClient, groupName);
        System.out.println(instanceId);
        autoScalingClient.close();
    }
}
```

```

    }

    public static String getAutoScaling(AutoScalingClient autoScalingClient, String
groupName) {
        try {
            String instanceId = "";
            DescribeAutoScalingGroupsRequest scalingGroupsRequest =
DescribeAutoScalingGroupsRequest.builder()
                .autoScalingGroupNames(groupName)
                .build();

            DescribeAutoScalingGroupsResponse response = autoScalingClient
                .describeAutoScalingGroups(scalingGroupsRequest);
            List<AutoScalingGroup> groups = response.autoScalingGroups();
            for (AutoScalingGroup group : groups) {
                System.out.println("The group name is " +
group.autoScalingGroupName());
                System.out.println("The group ARN is " +
group.autoScalingGroupARN());

                List<Instance> instances = group.instances();
                for (Instance instance : instances) {
                    instanceId = instance.instanceId();
                }
            }
            return instanceId;
        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
}

```

- Per i dettagli sull'API, consulta la [DescribeAutoScalingGroups](#) sezione AWS SDK for Java 2.x API Reference.

## DescribeAutoScalingInstances

Il seguente esempio di codice mostra come utilizzare `DescribeAutoScalingInstances`.

## SDK per Java 2.x

 Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public static void describeAutoScalingInstance(AutoScalingClient
autoScalingClient, String id) {
    try {
        DescribeAutoScalingInstancesRequest describeAutoScalingInstancesRequest
= DescribeAutoScalingInstancesRequest
        .builder()
        .instanceIds(id)
        .build();

        DescribeAutoScalingInstancesResponse response = autoScalingClient
        .describeAutoScalingInstances(describeAutoScalingInstancesRequest);
        List<AutoScalingInstanceDetails> instances =
response.autoScalingInstances();
        for (AutoScalingInstanceDetails instance : instances) {
            System.out.println("The instance lifecycle state is: " +
instance.lifecycleState());
        }

    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Per i dettagli sull'API, consulta la [DescribeAutoScalingInstances](#) sezione AWS SDK for Java 2.x API Reference.

## DescribeScalingActivities

Il seguente esempio di codice mostra come utilizzare `DescribeScalingActivities`.

## SDK per Java 2.x

 Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public static void describeScalingActivities(AutoScalingClient
autoScalingClient, String groupName) {
    try {
        DescribeScalingActivitiesRequest scalingActivitiesRequest =
DescribeScalingActivitiesRequest.builder()
            .autoScalingGroupName(groupName)
            .maxRecords(10)
            .build();

        DescribeScalingActivitiesResponse response = autoScalingClient
            .describeScalingActivities(scalingActivitiesRequest);
        List<Activity> activities = response.activities();
        for (Activity activity : activities) {
            System.out.println("The activity Id is " + activity.activityId());
            System.out.println("The activity details are " +
activity.details());
        }

    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Per i dettagli sull'API, consulta la [DescribeScalingActivities](#) sezione AWS SDK for Java 2.x API Reference.

## DisableMetricsCollection

Il seguente esempio di codice mostra come utilizzare `DisableMetricsCollection`.



## SDK per Java 2.x

 Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public static void disableMetricsCollection(AutoScalingClient autoScalingClient,
String groupName) {
    try {
        DisableMetricsCollectionRequest disableMetricsCollectionRequest =
DisableMetricsCollectionRequest.builder()
            .autoScalingGroupName(groupName)
            .metrics("GroupMaxSize")
            .build();

autoScalingClient.disableMetricsCollection(disableMetricsCollectionRequest);
        System.out.println("The disable metrics collection operation was
successful");

    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Per i dettagli sull'API, consulta la [DisableMetricsCollection](#) sezione AWS SDK for Java 2.x API Reference.

## EnableMetricsCollection

Il seguente esempio di codice mostra come utilizzare `EnableMetricsCollection`.

## SDK per Java 2.x

 Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public static void enableMetricsCollection(AutoScalingClient autoScalingClient,
String groupName) {
    try {
        EnableMetricsCollectionRequest collectionRequest =
EnableMetricsCollectionRequest.builder()
            .autoScalingGroupName(groupName)
            .metrics("GroupMaxSize")
            .granularity("1Minute")
            .build();

        autoScalingClient.enableMetricsCollection(collectionRequest);
        System.out.println("The enable metrics collection operation was
successful");
    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Per i dettagli sull'API, consulta la [EnableMetricsCollection](#) sezione AWS SDK for Java 2.x API Reference.

## SetDesiredCapacity

Il seguente esempio di codice mostra come utilizzare `SetDesiredCapacity`.

## SDK per Java 2.x

**Note**

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public static void setDesiredCapacity(AutoScalingClient autoScalingClient,
String groupName) {
    try {
        SetDesiredCapacityRequest capacityRequest =
SetDesiredCapacityRequest.builder()
            .autoScalingGroupName(groupName)
            .desiredCapacity(2)
            .build();

        autoScalingClient.setDesiredCapacity(capacityRequest);
        System.out.println("You have set the DesiredCapacity to 2");

    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Per i dettagli sull'API, consulta la [SetDesiredCapacity](#) sezione AWS SDK for Java 2.x API Reference.

**TerminateInstanceInAutoScalingGroup**

Il seguente esempio di codice mostra come utilizzare `TerminateInstanceInAutoScalingGroup`.

## SDK per Java 2.x

**Note**

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public static void terminateInstanceInAutoScalingGroup(AutoScalingClient
autoScalingClient, String instanceId) {
    try {
        TerminateInstanceInAutoScalingGroupRequest request =
        TerminateInstanceInAutoScalingGroupRequest.builder()
            .instanceId(instanceId)
            .shouldDecrementDesiredCapacity(false)
            .build();

        autoScalingClient.terminateInstanceInAutoScalingGroup(request);
        System.out.println("You have terminated instance " + instanceId);

    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Per i dettagli sull'API, consulta la [TerminateInstanceInAutoScalingGroup](#) sezione AWS SDK for Java 2.x API Reference.

## UpdateAutoScalingGroup

Il seguente esempio di codice mostra come utilizzare `UpdateAutoScalingGroup`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public static void updateAutoScalingGroup(AutoScalingClient autoScalingClient,
String groupName,
    String launchTemplateName) {
    try {
        AutoScalingWaiter waiter = autoScalingClient.waiter();
        LaunchTemplateSpecification templateSpecification =
        LaunchTemplateSpecification.builder()
```

```

        .launchTemplateName(launchTemplateName)
        .build();

        UpdateAutoScalingGroupRequest groupRequest =
UpdateAutoScalingGroupRequest.builder()
        .maxSize(3)
        .autoScalingGroupName(groupName)
        .launchTemplate(templateSpecification)
        .build();

        autoScalingClient.updateAutoScalingGroup(groupRequest);
        DescribeAutoScalingGroupsRequest groupsRequest =
DescribeAutoScalingGroupsRequest.builder()
        .autoScalingGroupNames(groupName)
        .build();

        WaiterResponse<DescribeAutoScalingGroupsResponse> waiterResponse =
waiter
        .waitUntilGroupInService(groupsRequest);
        waiterResponse.matched().response().ifPresent(System.out::println);
        System.out.println("You successfully updated the auto scaling group " +
groupName);

    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

```

- Per i dettagli sull'API, consulta la [UpdateAutoScalingGroup](#) sezione AWS SDK for Java 2.x API Reference.

## Scenari

### Creazione e gestione di un servizio resiliente

Il seguente esempio di codice mostra come creare un servizio Web con bilanciamento del carico che restituisca consigli su libri, film e canzoni. L'esempio mostra come il servizio risponde ai guasti e spiega come ristrutturarlo per una maggiore resilienza in caso di guasti.

- Utilizza un gruppo con dimensionamento automatico Amazon EC2 per creare istanze Amazon Elastic Compute Cloud (Amazon EC2) basate su un modello di avvio e per mantenere il numero di istanze entro un intervallo specificato.
- Gestisci e distribuisce le richieste HTTP con Elastic Load Balancing.
- Monitora lo stato delle istanze in un gruppo con dimensionamento automatico e inoltra le richieste soltanto alle istanze integre.
- Esegui un server Web Python su ogni istanza EC2 per gestire le richieste HTTP. Il server Web risponde con consigli e controlli dell'integrità.
- Simula un servizio di raccomandazione con una tabella Amazon DynamoDB.
- Controlla la risposta del server web alle richieste e ai controlli sanitari aggiornando AWS Systems Manager i parametri.

## SDK per Java 2.x

### Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Esegui lo scenario interattivo al prompt dei comandi.

```
public class Main {  
  
    public static final String fileName = "C:\\AWS\\resworkflow\\  
\\recommendations.json"; // Modify file location.  
    public static final String tableName = "doc-example-recommendation-service";  
    public static final String startScript = "C:\\AWS\\resworkflow\\  
\\server_startup_script.sh"; // Modify file location.  
    public static final String policyFile = "C:\\AWS\\resworkflow\\  
\\instance_policy.json"; // Modify file location.  
    public static final String ssmJSON = "C:\\AWS\\resworkflow\\  
\\ssm_only_policy.json"; // Modify file location.  
    public static final String failureResponse = "doc-example-resilient-  
architecture-failure-response";  
    public static final String healthCheck = "doc-example-resilient-architecture-  
health-check";  
    public static final String templateName = "doc-example-resilience-template";  
    public static final String roleName = "doc-example-resilience-role";  
}
```

```
public static final String policyName = "doc-example-resilience-pol";
public static final String profileName = "doc-example-resilience-prof";

public static final String badCredsProfileName = "doc-example-resilience-prof-
bc";

public static final String targetGroupName = "doc-example-resilience-tg";
public static final String autoScalingGroupName = "doc-example-resilience-
group";
public static final String lbName = "doc-example-resilience-lb";
public static final String protocol = "HTTP";
public static final int port = 80;

public static final String DASHES = new String(new char[80]).replace("\0", "-");

public static void main(String[] args) throws IOException, InterruptedException
{
    Scanner in = new Scanner(System.in);
    Database database = new Database();
    AutoScaler autoScaler = new AutoScaler();
    LoadBalancer loadBalancer = new LoadBalancer();

    System.out.println(DASHES);
    System.out.println("Welcome to the demonstration of How to Build and Manage
a Resilient Service!");
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("A - SETUP THE RESOURCES");
    System.out.println("Press Enter when you're ready to start deploying
resources.");
    in.nextLine();
    deploy(loadBalancer);
    System.out.println(DASHES);
    System.out.println(DASHES);
    System.out.println("B - DEMO THE RESILIENCE FUNCTIONALITY");
    System.out.println("Press Enter when you're ready.");
    in.nextLine();
    demo(loadBalancer);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("C - DELETE THE RESOURCES");
    System.out.println(""
```

This concludes the demo of how to build and manage a resilient service.

To keep things tidy and to avoid unwanted charges on your account, we can clean up all AWS resources

that were created for this demo.

```
""");
```

```
System.out.println("\n Do you want to delete the resources (y/n)? ");
```

```
String userInput = in.nextLine().trim().toLowerCase(); // Capture user input
```

```
if (userInput.equals("y")) {
```

```
    // Delete resources here
```

```
    deleteResources(loadBalancer, autoScaler, database);
```

```
    System.out.println("Resources deleted.");
```

```
} else {
```

```
    System.out.println("""
```

```
        Okay, we'll leave the resources intact.
```

```
        Don't forget to delete them when you're done with them or you
```

might incur unexpected charges.

```
""");
```

```
}
```

```
System.out.println(DASHES);
```

```
System.out.println(DASHES);
```

```
System.out.println("The example has completed. ");
```

```
System.out.println("\n Thanks for watching!");
```

```
System.out.println(DASHES);
```

```
}
```

```
// Deletes the AWS resources used in this example.
```

```
private static void deleteResources(LoadBalancer loadBalancer, AutoScaler autoScaler, Database database)
```

```
    throws IOException, InterruptedException {
```

```
    loadBalancer.deleteLoadBalancer(lbName);
```

```
    System.out.println("*** Wait 30 secs for resource to be deleted");
```

```
    TimeUnit.SECONDS.sleep(30);
```

```
    loadBalancer.deleteTargetGroup(targetGroupName);
```

```
    autoScaler.deleteAutoScaleGroup(autoScalingGroupName);
```

```
    autoScaler.deleteRolesPolicies(policyName, roleName, profileName);
```

```
    autoScaler.deleteTemplate(templateName);
```

```
    database.deleteTable(tableName);
```

```
}
```



```

private static void deploy(LoadBalancer loadBalancer) throws
InterruptedException, IOException {
    Scanner in = new Scanner(System.in);
    System.out.println(
        """
                For this demo, we'll use the AWS SDK for Java (v2) to create
several AWS resources
                to set up a load-balanced web service endpoint and explore
some ways to make it resilient
                against various kinds of failures.

                Some of the resources create by this demo are:
                \t* A DynamoDB table that the web service depends on to
provide book, movie, and song recommendations.
                \t* An EC2 launch template that defines EC2 instances that
each contain a Python web server.
                \t* An EC2 Auto Scaling group that manages EC2 instances
across several Availability Zones.
                \t* An Elastic Load Balancing (ELB) load balancer that
targets the Auto Scaling group to distribute requests.
        """);

    System.out.println("Press Enter when you're ready.");
    in.nextLine();
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("Creating and populating a DynamoDB table named " +
tableName);
    Database database = new Database();
    database.createTable(tableName, fileName);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("""
                Creating an EC2 launch template that runs '{startup_script}' when an
instance starts.
                This script starts a Python web server defined in the `server.py`
script. The web server
                listens to HTTP requests on port 80 and responds to requests to '/'
and to '/healthcheck'.
                For demo purposes, this server is run as the root user. In
production, the best practice is to
                run a web server, such as Apache, with least-privileged credentials.
    """);

```

```
        The template also defines an IAM policy that each instance uses to
assume a role that grants
        permissions to access the DynamoDB recommendation table and Systems
Manager parameters
        that control the flow of the demo.
        """);
```

```
        LaunchTemplateCreator templateCreator = new LaunchTemplateCreator();
        templateCreator.createTemplate(policyFile, policyName, profileName,
startScript, templateName, roleName);
        System.out.println(DASHES);
```

```
        System.out.println(DASHES);
        System.out.println(
            "Creating an EC2 Auto Scaling group that maintains three EC2
instances, each in a different Availability Zone.");
        System.out.println("*** Wait 30 secs for the VPC to be created");
        TimeUnit.SECONDS.sleep(30);
        AutoScaler autoScaler = new AutoScaler();
        String[] zones = autoScaler.createGroup(3, templateName,
autoScalingGroupName);
```

```
        System.out.println("""
            At this point, you have EC2 instances created. Once each instance
starts, it listens for
            HTTP requests. You can see these instances in the console or
continue with the demo.
            Press Enter when you're ready to continue.
            """);
```

```
        in.nextLine();
        System.out.println(DASHES);
```

```
        System.out.println(DASHES);
        System.out.println("Creating variables that control the flow of the demo.");
        ParameterHelper paramHelper = new ParameterHelper();
        paramHelper.reset();
        System.out.println(DASHES);
```

```
        System.out.println(DASHES);
        System.out.println("""
            Creating an Elastic Load Balancing target group and load balancer.
The target group
```

```

        defines how the load balancer connects to instances. The load
balancer provides a
        single endpoint where clients connect and dispatches requests to
instances in the group.
        """);

    String vpcId = autoScaler.getDefaultVPC();
    List<Subnet> subnets = autoScaler.getSubnets(vpcId, zones);
    System.out.println("You have retrieved a list with " + subnets.size() + "
subnets");
    String targetGroupArn = loadBalancer.createTargetGroup(protocol, port,
vpcId, targetGroupName);
    String elbDnsName = loadBalancer.createLoadBalancer(subnets, targetGroupArn,
lbName, port, protocol);
    autoScaler.attachLoadBalancerTargetGroup(autoScalingGroupName,
targetGroupArn);
    System.out.println("Verifying access to the load balancer endpoint...");
    boolean wasSuccessful = loadBalancer.verifyLoadBalancerEndpoint(elbDnsName);
    if (!wasSuccessful) {
        System.out.println("Couldn't connect to the load balancer, verifying
that the port is open...");
        CloseableHttpClient httpClient = HttpClients.createDefault();

        // Create an HTTP GET request to "http://checkip.amazonaws.com"
        HttpGet httpGet = new HttpGet("http://checkip.amazonaws.com");
        try {
            // Execute the request and get the response
            HttpResponse response = httpClient.execute(httpGet);

            // Read the response content.
            String ipAddress =
IOUtils.toString(response.getEntity().getContent(), StandardCharsets.UTF_8).trim();

            // Print the public IP address.
            System.out.println("Public IP Address: " + ipAddress);
            GroupInfo groupInfo = autoScaler.verifyInboundPort(vpcId, port,
ipAddress);
            if (!groupInfo.isPortOpen()) {
                System.out.println("""
                    For this example to work, the default security group for
your default VPC must
                    allow access from this computer. You can either add it
automatically from this

```

```

        example or add it yourself using the AWS Management
Console.
        """);

        System.out.println(
            "Do you want to add a rule to security group " +
groupInfo.getGroupName() + " to allow");
        System.out.println("inbound traffic on port " + port + " from
your computer's IP address (y/n) ");
        String ans = in.nextLine();
        if ("y".equalsIgnoreCase(ans)) {
            autoScaler.openInboundPort(groupInfo.getGroupName(),
String.valueOf(port), ipAddress);
            System.out.println("Security group rule added.");
        } else {
            System.out.println("No security group rule added.");
        }
    }

    } catch (AutoScalingException e) {
        e.printStackTrace();
    }
    } else if (wasSuccessful) {
        System.out.println("Your load balancer is ready. You can access it by
browsing to:");
        System.out.println("\t http://" + elbDnsName);
    } else {
        System.out.println("Couldn't get a successful response from the load
balancer endpoint. Troubleshoot by");
        System.out.println("manually verifying that your VPC and security group
are configured correctly and that");
        System.out.println("you can successfully make a GET request to the load
balancer.");
    }

    System.out.println("Press Enter when you're ready to continue with the
demo.");
    in.nextLine();
}

// A method that controls the demo part of the Java program.
public static void demo(LoadBalancer loadBalancer) throws IOException,
InterruptedException {
    ParameterHelper paramHelper = new ParameterHelper();

```

```
System.out.println("Read the ssm_only_policy.json file");
String ssmOnlyPolicy = readFileAsString(ssmJSON);

System.out.println("Resetting parameters to starting values for demo.");
paramHelper.reset();

System.out.println(
    """
        This part of the demonstration shows how to toggle
different parts of the system
        to create situations where the web service fails, and shows
how using a resilient
        architecture can keep the web service running in spite of
these failures.

        At the start, the load balancer endpoint returns
recommendations and reports that all targets are healthy.
    """);
demoChoices(loadBalancer);

System.out.println(
    """
        The web service running on the EC2 instances gets
recommendations by querying a DynamoDB table.
        The table name is contained in a Systems Manager parameter
named self.param_helper.table.
        To simulate a failure of the recommendation service, let's
set this parameter to name a non-existent table.
    """);
paramHelper.put(paramHelper.tableName, "this-is-not-a-table");

System.out.println(
    """
        \nNow, sending a GET request to the load balancer endpoint
returns a failure code. But, the service reports as
        healthy to the load balancer because shallow health checks
don't check for failure of the recommendation service.
    """);
demoChoices(loadBalancer);

System.out.println(
    """
        Instead of failing when the recommendation service fails,
the web service can return a static response.
    """);
```

```

        While this is not a perfect solution, it presents the
customer with a somewhat better experience than failure.
        """);
    paramHelper.put(paramHelper.failureResponse, "static");

    System.out.println("""
        Now, sending a GET request to the load balancer endpoint returns a
static response.
        The service still reports as healthy because health checks are still
shallow.
        """);
    demoChoices(loadBalancer);

    System.out.println("Let's reinstate the recommendation service.");
    paramHelper.put(paramHelper.tableName, paramHelper.dyntable);

    System.out.println("""
        Let's also substitute bad credentials for one of the instances in
the target group so that it can't
        access the DynamoDB recommendation table. We will get an instance id
value.
        """);

    LaunchTemplateCreator templateCreator = new LaunchTemplateCreator();
    AutoScaler autoScaler = new AutoScaler();

    // Create a new instance profile based on badCredsProfileName.
    templateCreator.createInstanceProfile(policyFile, policyName,
badCredsProfileName, roleName);
    String badInstanceId = autoScaler.getBadInstance(autoScalingGroupName);
    System.out.println("The bad instance id values used for this demo is " +
badInstanceId);

    String profileAssociationId = autoScaler.getInstanceProfile(badInstanceId);
    System.out.println("The association Id value is " + profileAssociationId);
    System.out.println("Replacing the profile for instance " + badInstanceId
        + " with a profile that contains bad credentials");
    autoScaler.replaceInstanceProfile(badInstanceId, badCredsProfileName,
profileAssociationId);

    System.out.println(
        """"
        Now, sending a GET request to the load balancer endpoint
returns either a recommendation or a static response,

```

```
        depending on which instance is selected by the load
balancer.
        """);

    demoChoices(loadBalancer);

    System.out.println("""
        Let's implement a deep health check. For this demo, a deep health
check tests whether
        the web service can access the DynamoDB table that it depends on for
recommendations. Note that
        the deep health check is only for ELB routing and not for Auto
Scaling instance health.
        This kind of deep health check is not recommended for Auto Scaling
instance health, because it
        risks accidental termination of all instances in the Auto Scaling
group when a dependent service fails.
        """);

    System.out.println("""
        By implementing deep health checks, the load balancer can detect
when one of the instances is failing
        and take that instance out of rotation.
        """);

    paramHelper.put(paramHelper.healthCheck, "deep");

    System.out.println("""
        Now, checking target health indicates that the instance with bad
credentials
        is unhealthy. Note that it might take a minute or two for the load
balancer to detect the unhealthy
        instance. Sending a GET request to the load balancer endpoint always
returns a recommendation, because
        the load balancer takes unhealthy instances out of its rotation.
        """);

    demoChoices(loadBalancer);

    System.out.println(
        ""
        Because the instances in this demo are controlled by an auto
scaler, the simplest way to fix an unhealthy
```

```

        instance is to terminate it and let the auto scaler start a
new instance to replace it.
        """);
        autoScaler.terminateInstance(badInstanceId);

        System.out.println("""
            Even while the instance is terminating and the new instance is
starting, sending a GET
            request to the web service continues to get a successful
recommendation response because
            the load balancer routes requests to the healthy instances. After
the replacement instance
            starts and reports as healthy, it is included in the load balancing
rotation.

            Note that terminating and replacing an instance typically takes
several minutes, during which time you
            can see the changing health check status until the new instance is
running and healthy.
        """);

        demoChoices(loadBalancer);
        System.out.println(
            "If the recommendation service fails now, deep health checks mean
all instances report as unhealthy.");
        paramHelper.put(paramHelper.tableName, "this-is-not-a-table");

        demoChoices(loadBalancer);
        paramHelper.reset();
    }

    public static void demoChoices(LoadBalancer loadBalancer) throws IOException,
InterruptedException {
        String[] actions = {
            "Send a GET request to the load balancer endpoint.",
            "Check the health of load balancer targets.",
            "Go to the next part of the demo."
        };
        Scanner scanner = new Scanner(System.in);

        while (true) {
            System.out.println("-".repeat(88));
            System.out.println("See the current state of the service by selecting
one of the following choices:");
            for (int i = 0; i < actions.length; i++) {

```



```
        System.out.println(i + ": " + actions[i]);
    }

    try {
        System.out.print("\nWhich action would you like to take? ");
        int choice = scanner.nextInt();
        System.out.println("-".repeat(88));

        switch (choice) {
            case 0 -> {
                System.out.println("Request:\n");
                System.out.println("GET http://" +
loadBalancer.getEndpoint(lbName));
                CloseableHttpClient httpClient =
HttpClientClients.createDefault();

                // Create an HTTP GET request to the ELB.
                HttpGet httpGet = new HttpGet("http://" +
loadBalancer.getEndpoint(lbName));

                // Execute the request and get the response.
                HttpResponse response = httpClient.execute(httpGet);
                int statusCode = response.getStatusLine().getStatusCode();
                System.out.println("HTTP Status Code: " + statusCode);

                // Display the JSON response
                BufferedReader reader = new BufferedReader(
                    new
InputStreamReader(response.getEntity().getContent()));
                StringBuilder jsonResponse = new StringBuilder();
                String line;
                while ((line = reader.readLine()) != null) {
                    jsonResponse.append(line);
                }
                reader.close();

                // Print the formatted JSON response.
                System.out.println("Full Response:\n");
                System.out.println(jsonResponse.toString());

                // Close the HTTP client.
                httpClient.close();
            }
        }
    }
```

```

        case 1 -> {
            System.out.println("\nChecking the health of load balancer
targets:\n");
            List<TargetHealthDescription> health =
loadBalancer.checkTargetHealth(targetGroupName);
            for (TargetHealthDescription target : health) {
                System.out.printf("\tTarget %s on port %d is %s\n",
target.target().id(),
                                target.target().port(),
target.targetHealth().stateAsString());
            }
            System.out.println("""
Note that it can take a minute or two for the health
check to update
                                after changes are made.
                                """);
        }
        case 2 -> {
            System.out.println("\nOkay, let's move on.");
            System.out.println("-".repeat(88));
            return; // Exit the method when choice is 2
        }
        default -> System.out.println("You must choose a value between
0-2. Please select again.");
    }

    } catch (java.util.InputMismatchException e) {
        System.out.println("Invalid input. Please select again.");
        scanner.nextLine(); // Clear the input buffer.
    }
}

public static String readFileAsString(String filePath) throws IOException {
    byte[] bytes = Files.readAllBytes(Paths.get(filePath));
    return new String(bytes);
}
}

```

Crea una classe che racchiuda le operazioni di dimensionamento automatico e Amazon EC2.

```
public class AutoScaler {
```

```
private static Ec2Client ec2Client;
private static AutoScalingClient autoScalingClient;
private static IamClient iamClient;

private static SsmClient ssmClient;

private IamClient getIAMClient() {
    if (iamClient == null) {
        iamClient = IamClient.builder()
            .region(Region.US_EAST_1)
            .build();
    }
    return iamClient;
}

private SsmClient getSSMClient() {
    if (ssmClient == null) {
        ssmClient = SsmClient.builder()
            .region(Region.US_EAST_1)
            .build();
    }
    return ssmClient;
}

private Ec2Client getEc2Client() {
    if (ec2Client == null) {
        ec2Client = Ec2Client.builder()
            .region(Region.US_EAST_1)
            .build();
    }
    return ec2Client;
}

private AutoScalingClient getAutoScalingClient() {
    if (autoScalingClient == null) {
        autoScalingClient = AutoScalingClient.builder()
            .region(Region.US_EAST_1)
            .build();
    }
    return autoScalingClient;
}

/**
```

```
    * Terminates and instances in an EC2 Auto Scaling group. After an instance is
    * terminated, it can no longer be accessed.
    */
    public void terminateInstance(String instanceId) {
        TerminateInstanceInAutoScalingGroupRequest terminateInstanceIRequest =
        TerminateInstanceInAutoScalingGroupRequest
            .builder()
            .instanceId(instanceId)
            .shouldDecrementDesiredCapacity(false)
            .build();

        getAutoScalingClient().terminateInstanceInAutoScalingGroup(terminateInstanceIRequest);
        System.out.format("Terminated instance %s.", instanceId);
    }

    /**
     * Replaces the profile associated with a running instance. After the profile is
     * replaced, the instance is rebooted to ensure that it uses the new profile.
     * When
     * the instance is ready, Systems Manager is used to restart the Python web
     * server.
     */
    public void replaceInstanceProfile(String instanceId, String
    newInstanceProfileName, String profileAssociationId)
        throws InterruptedException {
        // Create an IAM instance profile specification.
        software.amazon.awssdk.services.ec2.model.IamInstanceProfileSpecification
        iamInstanceProfile =
        software.amazon.awssdk.services.ec2.model.IamInstanceProfileSpecification
            .builder()
            .name(newInstanceProfileName) // Make sure 'newInstanceProfileName'
is a valid IAM Instance Profile
            // name.
            .build();

        // Replace the IAM instance profile association for the EC2 instance.
        ReplaceIamInstanceProfileAssociationRequest replaceRequest =
        ReplaceIamInstanceProfileAssociationRequest
            .builder()
            .iamInstanceProfile(iamInstanceProfile)
            .associationId(profileAssociationId) // Make sure
'profileAssociationId' is a valid association ID.
            .build();
    }
}
```

```
try {
    getEc2Client().replaceIamInstanceProfileAssociation(replaceRequest);
    // Handle the response as needed.
} catch (Ec2Exception e) {
    // Handle exceptions, log, or report the error.
    System.err.println("Error: " + e.getMessage());
}
System.out.format("Replaced instance profile for association %s with profile
%s.", profileAssociationId,
    newInstanceProfileName);
TimeUnit.SECONDS.sleep(15);
boolean instReady = false;
int tries = 0;

// Reboot after 60 seconds
while (!instReady) {
    if (tries % 6 == 0) {
        getEc2Client().rebootInstances(RebootInstancesRequest.builder()
            .instanceIds(instanceId)
            .build());
        System.out.println("Rebooting instance " + instanceId + " and
waiting for it to be ready.");
    }
    tries++;
    try {
        TimeUnit.SECONDS.sleep(10);
    } catch (InterruptedException e) {
        e.printStackTrace();
    }

    DescribeInstanceInformationResponse informationResponse =
getSSMClient().describeInstanceInformation();
    List<InstanceInformation> instanceInformationList =
informationResponse.getInstanceInformationList();
    for (InstanceInformation info : instanceInformationList) {
        if (info.getInstanceId().equals(instanceId)) {
            instReady = true;
            break;
        }
    }
}

SendCommandRequest sendCommandRequest = SendCommandRequest.builder()
```

```

        .instanceIds(instanceId)
        .documentName("AWS-RunShellScript")
        .parameters(Collections.singletonMap("commands",
            Collections.singletonList("cd / && sudo python3 server.py
80")))
        .build();

        getSSMClient().sendCommand(sendCommandRequest);
        System.out.println("Restarted the Python web server on instance " +
instanceId + ".");
    }

    public void openInboundPort(String secGroupId, String port, String ipAddress) {
        AuthorizeSecurityGroupIngressRequest ingressRequest =
AuthorizeSecurityGroupIngressRequest.builder()
            .groupName(secGroupId)
            .cidrIp(ipAddress)
            .fromPort(Integer.parseInt(port))
            .build();

        getEc2Client().authorizeSecurityGroupIngress(ingressRequest);
        System.out.format("Authorized ingress to %s on port %s from %s.",
secGroupId, port, ipAddress);
    }

    /**
     * Detaches a role from an instance profile, detaches policies from the role,
     * and deletes all the resources.
     */
    public void deleteInstanceProfile(String roleName, String profileName) {
        try {
            software.amazon.awssdk.services.iam.model.GetInstanceProfileRequest
getInstanceProfileRequest =
software.amazon.awssdk.services.iam.model.GetInstanceProfileRequest
            .builder()
            .instanceProfileName(profileName)
            .build();

            GetInstanceProfileResponse response =
getIAMClient().getInstanceProfile(getInstanceProfileRequest);
            String name = response.getInstanceProfile().getInstanceProfileName();
            System.out.println(name);
        }
    }

```

```
        RemoveRoleFromInstanceProfileRequest profileRequest =
RemoveRoleFromInstanceProfileRequest.builder()
    .instanceProfileName(profileName)
    .roleName(roleName)
    .build();

        getIAMClient().removeRoleFromInstanceProfile(profileRequest);
        DeleteInstanceProfileRequest deleteInstanceProfileRequest =
DeleteInstanceProfileRequest.builder()
    .instanceProfileName(profileName)
    .build();

        getIAMClient().deleteInstanceProfile(deleteInstanceProfileRequest);
        System.out.println("Deleted instance profile " + profileName);

        DeleteRoleRequest deleteRoleRequest = DeleteRoleRequest.builder()
    .roleName(roleName)
    .build();

        // List attached role policies.
        ListAttachedRolePoliciesResponse rolesResponse = getIAMClient()
    .listAttachedRolePolicies(role -> role.roleName(roleName));
        List<AttachedPolicy> attachedPolicies =
rolesResponse.attachedPolicies();
        for (AttachedPolicy attachedPolicy : attachedPolicies) {
            DetachRolePolicyRequest request = DetachRolePolicyRequest.builder()
    .roleName(roleName)
    .policyArn(attachedPolicy.policyArn())
    .build();

            getIAMClient().detachRolePolicy(request);
            System.out.println("Detached and deleted policy " +
attachedPolicy.policyName());
        }

        getIAMClient().deleteRole(deleteRoleRequest);
        System.out.println("Instance profile and role deleted.");

    } catch (IamException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

```
public void deleteTemplate(String templateName) {
    getEc2Client().deleteLaunchTemplate(name ->
name.launchTemplateName(templateName));
    System.out.format(templateName + " was deleted.");
}

public void deleteAutoScaleGroup(String groupName) {
    DeleteAutoScalingGroupRequest deleteAutoScalingGroupRequest =
DeleteAutoScalingGroupRequest.builder()
        .autoScalingGroupName(groupName)
        .forceDelete(true)
        .build();

getAutoScalingClient().deleteAutoScalingGroup(deleteAutoScalingGroupRequest);
    System.out.println(groupName + " was deleted.");
}

/*
 * Verify the default security group of the specified VPC allows ingress from
 * this
 * computer. This can be done by allowing ingress from this computer's IP
 * address. In some situations, such as connecting from a corporate network, you
 * must instead specify a prefix list ID. You can also temporarily open the port
 * to
 * any IP address while running this example. If you do, be sure to remove
 * public
 * access when you're done.
 */
public GroupInfo verifyInboundPort(String VPC, int port, String ipAddress) {
    boolean portIsOpen = false;
    GroupInfo groupInfo = new GroupInfo();
    try {
        Filter filter = Filter.builder()
            .name("group-name")
            .values("default")
            .build();

        Filter filter1 = Filter.builder()
            .name("vpc-id")
            .values(VPC)
            .build();
```



```

        DescribeSecurityGroupsRequest securityGroupsRequest =
DescribeSecurityGroupsRequest.builder()
        .filters(filter, filter1)
        .build();

        DescribeSecurityGroupsResponse securityGroupsResponse = getEc2Client()
        .describeSecurityGroups(securityGroupsRequest);

        String securityGroup =
securityGroupsResponse.securityGroups().get(0).groupName();
        groupInfo.setGroupName(securityGroup);

        for (SecurityGroup secGroup : securityGroupsResponse.securityGroups()) {
            System.out.println("Found security group: " + secGroup.groupId());

            for (IpPermission ipPermission : secGroup.ipPermissions()) {
                if (ipPermission.fromPort() == port) {
                    System.out.println("Found inbound rule: " + ipPermission);
                    for (IpRange ipRange : ipPermission.ipRanges()) {
                        String cidrIp = ipRange.cidrIp();
                        if (cidrIp.startsWith(ipAddress) ||
cidrIp.equals("0.0.0.0/0")) {
                            System.out.println(cidrIp + " is applicable");
                            portIsOpen = true;
                        }
                    }

                    if (!ipPermission.prefixListIds().isEmpty()) {
                        System.out.println("Prefix lList is applicable");
                        portIsOpen = true;
                    }

                    if (!portIsOpen) {
                        System.out
                            .println("The inbound rule does not appear to be
open to either this computer's IP,"
                                + " all IP addresses (0.0.0.0/0), or to
a prefix list ID.");
                    } else {
                        break;
                    }
                }
            }
        }
    }
}

```

```
    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }

    groupInfo.setPortOpen(portIsOpen);
    return groupInfo;
}

/**
 * Attaches an Elastic Load Balancing (ELB) target group to this EC2 Auto
 * Scaling group.
 * The target group specifies how the load balancer forward requests to the
 * instances
 * in the group.
 */
public void attachLoadBalancerTargetGroup(String asGroupName, String
targetGroupARN) {
    try {
        AttachLoadBalancerTargetGroupsRequest targetGroupsRequest =
AttachLoadBalancerTargetGroupsRequest.builder()
            .autoScalingGroupName(asGroupName)
            .targetGroupARNs(targetGroupARN)
            .build();

getAutoScalingClient().attachLoadBalancerTargetGroups(targetGroupsRequest);
        System.out.println("Attached load balancer to " + asGroupName);

    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

// Creates an EC2 Auto Scaling group with the specified size.
public String[] createGroup(int groupSize, String templateName, String
autoScalingGroupName) {

    // Get availability zones.
    software.amazon.awssdk.services.ec2.model.DescribeAvailabilityZonesRequest
zonesRequest =
software.amazon.awssdk.services.ec2.model.DescribeAvailabilityZonesRequest
            .builder()
            .build();
```

```
DescribeAvailabilityZonesResponse zonesResponse =
getEc2Client().describeAvailabilityZones(zonesRequest);
List<String> availabilityZoneNames =
zonesResponse.availabilityZones().stream()

.map(software.amazon.awssdk.services.ec2.model.AvailabilityZone::zoneName)
    .collect(Collectors.toList());

String availabilityZones = String.join(",", availabilityZoneNames);
LaunchTemplateSpecification specification =
LaunchTemplateSpecification.builder()
    .launchTemplateName(templateName)
    .version("$Default")
    .build();

String[] zones = availabilityZones.split(",");
CreateAutoScalingGroupRequest groupRequest =
CreateAutoScalingGroupRequest.builder()
    .launchTemplate(specification)
    .availabilityZones(zones)
    .maxSize(groupSize)
    .minSize(groupSize)
    .autoScalingGroupName(autoScalingGroupName)
    .build();

try {
    getAutoScalingClient().createAutoScalingGroup(groupRequest);
} catch (AutoScalingException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}

System.out.println("Created an EC2 Auto Scaling group named " +
autoScalingGroupName);
return zones;
}

public String getDefaultVPC() {
    // Define the filter.
    Filter defaultFilter = Filter.builder()
        .name("is-default")
        .values("true")
        .build();
```

```
        software.amazon.awssdk.services.ec2.model.DescribeVpcsRequest request =
software.amazon.awssdk.services.ec2.model.DescribeVpcsRequest
            .builder()
            .filters(defaultFilter)
            .build();

        DescribeVpcsResponse response = getEc2Client().describeVpcs(request);
        return response.vpcs().get(0).vpcId();
    }

    // Gets the default subnets in a VPC for a specified list of Availability Zones.
    public List<Subnet> getSubnets(String vpcId, String[] availabilityZones) {
        List<Subnet> subnets = null;
        Filter vpcFilter = Filter.builder()
            .name("vpc-id")
            .values(vpcId)
            .build();

        Filter azFilter = Filter.builder()
            .name("availability-zone")
            .values(availabilityZones)
            .build();

        Filter defaultForAZ = Filter.builder()
            .name("default-for-az")
            .values("true")
            .build();

        DescribeSubnetsRequest request = DescribeSubnetsRequest.builder()
            .filters(vpcFilter, azFilter, defaultForAZ)
            .build();

        DescribeSubnetsResponse response = getEc2Client().describeSubnets(request);
        subnets = response.subnets();
        return subnets;
    }

    // Gets data about the instances in the EC2 Auto Scaling group.
    public String getBadInstance(String groupName) {
        DescribeAutoScalingGroupsRequest request =
DescribeAutoScalingGroupsRequest.builder()
            .autoScalingGroupNames(groupName)
            .build();
```

```

        DescribeAutoScalingGroupsResponse response =
getAutoScalingClient().describeAutoScalingGroups(request);
        AutoScalingGroup autoScalingGroup = response.autoScalingGroups().get(0);
        List<String> instanceIds = autoScalingGroup.instances().stream()
                .map(instance -> instance.instanceId())
                .collect(Collectors.toList());

        String[] instanceIdArray = instanceIds.toArray(new String[0]);
        for (String instanceId : instanceIdArray) {
            System.out.println("Instance ID: " + instanceId);
            return instanceId;
        }
        return "";
    }

    // Gets data about the profile associated with an instance.
    public String getInstanceProfile(String instanceId) {
        Filter filter = Filter.builder()
                .name("instance-id")
                .values(instanceId)
                .build();

        DescribeIamInstanceProfileAssociationsRequest associationsRequest =
DescribeIamInstanceProfileAssociationsRequest
                .builder()
                .filters(filter)
                .build();

        DescribeIamInstanceProfileAssociationsResponse response = getEc2Client()
                .describeIamInstanceProfileAssociations(associationsRequest);
        return response.iamInstanceProfileAssociations().get(0).associationId();
    }

    public void deleteRolesPolicies(String policyName, String roleName, String
InstanceProfile) {
        ListPoliciesRequest listPoliciesRequest =
ListPoliciesRequest.builder().build();
        ListPoliciesResponse listPoliciesResponse =
getIAMClient().listPolicies(listPoliciesRequest);
        for (Policy policy : listPoliciesResponse.policies()) {
            if (policy.policyName().equals(policyName)) {
                // List the entities (users, groups, roles) that are attached to the
policy.

```

```

software.amazon.awssdk.services.iam.model.ListEntitiesForPolicyRequest
listEntitiesRequest =
software.amazon.awssdk.services.iam.model.ListEntitiesForPolicyRequest
    .builder()
    .policyArn(policy.arn())
    .build();
    ListEntitiesForPolicyResponse listEntitiesResponse = iamClient
        .listEntitiesForPolicy(listEntitiesRequest);
    if (!listEntitiesResponse.policyGroups().isEmpty() || !
listEntitiesResponse.policyUsers().isEmpty()
        || !listEntitiesResponse.policyRoles().isEmpty()) {
        // Detach the policy from any entities it is attached to.
        DetachRolePolicyRequest detachPolicyRequest =
DetachRolePolicyRequest.builder()
            .policyArn(policy.arn())
            .roleName(roleName) // Specify the name of the IAM role
            .build();

        getIAMClient().detachRolePolicy(detachPolicyRequest);
        System.out.println("Policy detached from entities.");
    }

    // Now, you can delete the policy.
    DeletePolicyRequest deletePolicyRequest =
DeletePolicyRequest.builder()
        .policyArn(policy.arn())
        .build();

    getIAMClient().deletePolicy(deletePolicyRequest);
    System.out.println("Policy deleted successfully.");
    break;
}
}

// List the roles associated with the instance profile
ListInstanceProfilesForRoleRequest listRolesRequest =
ListInstanceProfilesForRoleRequest.builder()
    .roleName(roleName)
    .build();

// Detach the roles from the instance profile
ListInstanceProfilesForRoleResponse listRolesResponse =
iamClient.listInstanceProfilesForRole(listRolesRequest);

```

```

        for (software.amazon.awssdk.services.iam.model.InstanceProfile profile :
listRolesResponse.instanceProfiles()) {
            RemoveRoleFromInstanceProfileRequest removeRoleRequest =
RemoveRoleFromInstanceProfileRequest.builder()
                .instanceProfileName(InstanceProfile)
                .roleName(roleName) // Remove the extra dot here
                .build();

            getIAMClient().removeRoleFromInstanceProfile(removeRoleRequest);
            System.out.println("Role " + roleName + " removed from instance profile
" + InstanceProfile);
        }

        // Delete the instance profile after removing all roles
        DeleteInstanceProfileRequest deleteInstanceProfileRequest =
DeleteInstanceProfileRequest.builder()
            .instanceProfileName(InstanceProfile)
            .build();

        getIAMClient().deleteInstanceProfile(r ->
r.instanceProfileName(InstanceProfile));
        System.out.println(InstanceProfile + " Deleted");
        System.out.println("All roles and policies are deleted.");
    }
}

```

Crea una classe che racchiuda le operazioni di Elastic Load Balancing.

```

public class LoadBalancer {
    public ElasticLoadBalancingV2Client elasticLoadBalancingV2Client;

    public ElasticLoadBalancingV2Client getLoadBalancerClient() {
        if (elasticLoadBalancingV2Client == null) {
            elasticLoadBalancingV2Client = ElasticLoadBalancingV2Client.builder()
                .region(Region.US_EAST_1)
                .build();
        }

        return elasticLoadBalancingV2Client;
    }

    // Checks the health of the instances in the target group.

```

```
public List<TargetHealthDescription> checkTargetHealth(String targetGroupName) {
    DescribeTargetGroupsRequest targetGroupsRequest =
DescribeTargetGroupsRequest.builder()
        .names(targetGroupName)
        .build();

    DescribeTargetGroupsResponse tgResponse =
getLoadBalancerClient().describeTargetGroups(targetGroupsRequest);

    DescribeTargetHealthRequest healthRequest =
DescribeTargetHealthRequest.builder()
        .targetGroupArn(tgResponse.targetGroups().get(0).targetGroupArn())
        .build();

    DescribeTargetHealthResponse healthResponse =
getLoadBalancerClient().describeTargetHealth(healthRequest);
    return healthResponse.targetHealthDescriptions();
}

// Gets the HTTP endpoint of the load balancer.
public String getEndpoint(String lbName) {
    DescribeLoadBalancersResponse res = getLoadBalancerClient()
        .describeLoadBalancers(describe -> describe.names(lbName));
    return res.loadBalancers().get(0).dnsName();
}

// Deletes a load balancer.
public void deleteLoadBalancer(String lbName) {
    try {
        // Use a waiter to delete the Load Balancer.
        DescribeLoadBalancersResponse res = getLoadBalancerClient()
            .describeLoadBalancers(describe -> describe.names(lbName));
        ElasticLoadBalancingV2Waiter loadBalancerWaiter =
getLoadBalancerClient().waiter();
        DescribeLoadBalancersRequest request =
DescribeLoadBalancersRequest.builder()
            .loadBalancerArns(res.loadBalancers().get(0).loadBalancerArn())
            .build();

        getLoadBalancerClient().deleteLoadBalancer(
            builder ->
builder.loadBalancerArn(res.loadBalancers().get(0).loadBalancerArn()));
        WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =
loadBalancerWaiter
```



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

    } catch (ElasticLoadBalancingV2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }
    System.out.println(lbName + " was deleted.");
}

// Deletes the target group.
public void deleteTargetGroup(String targetGroupName) {
    try {
        DescribeTargetGroupsResponse res = getLoadBalancerClient()
            .describeTargetGroups(describe ->
describe.names(targetGroupName));
        getLoadBalancerClient()
            .deleteTargetGroup(builder ->
builder.targetGroupArn(res.targetGroups().get(0).targetGroupArn()));
    } catch (ElasticLoadBalancingV2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }
    System.out.println(targetGroupName + " was deleted.");
}

// Verify this computer can successfully send a GET request to the load balancer
// endpoint.
public boolean verifyLoadBalancerEndpoint(String elbDnsName) throws IOException,
InterruptedException {
    boolean success = false;
    int retries = 3;
    CloseableHttpClient httpClient = HttpClients.createDefault();

    // Create an HTTP GET request to the ELB.
    HttpGet httpGet = new HttpGet("http://" + elbDnsName);
    try {
        while ((!success) && (retries > 0)) {
            // Execute the request and get the response.
            HttpResponse response = httpClient.execute(httpGet);
            int statusCode = response.getStatusLine().getStatusCode();
            System.out.println("HTTP Status Code: " + statusCode);
            if (statusCode == 200) {
                success = true;
            } else {
                retries--;
            }
        }
    }
}
```

```
        System.out.println("Got connection error from load balancer
endpoint, retrying...");
        TimeUnit.SECONDS.sleep(15);
    }
}

} catch (org.apache.http.conn.HttpHostConnectException e) {
    System.out.println(e.getMessage());
}

System.out.println("Status.." + success);
return success;
}

/*
 * Creates an Elastic Load Balancing target group. The target group specifies
 * how
 * the load balancer forward requests to instances in the group and how instance
 * health is checked.
 */
public String createTargetGroup(String protocol, int port, String vpcId, String
targetGroupName) {
    CreateTargetGroupRequest targetGroupRequest =
CreateTargetGroupRequest.builder()
        .healthCheckPath("/healthcheck")
        .healthCheckTimeoutSeconds(5)
        .port(port)
        .vpcId(vpcId)
        .name(targetGroupName)
        .protocol(protocol)
        .build();

    CreateTargetGroupResponse targetGroupResponse =
getLoadBalancerClient().createTargetGroup(targetGroupRequest);
    String targetGroupArn =
targetGroupResponse.targetGroups().get(0).targetGroupArn();
    String targetGroup =
targetGroupResponse.targetGroups().get(0).targetGroupName();
    System.out.println("The " + targetGroup + " was created with ARN" +
targetGroupArn);
    return targetGroupArn;
}

/*
```

```
* Creates an Elastic Load Balancing load balancer that uses the specified
* subnets
* and forwards requests to the specified target group.
*/
public String createLoadBalancer(List<Subnet> subnetIds, String targetGroupARN,
String lbName, int port,
    String protocol) {
    try {
        List<String> subnetIdStrings = subnetIds.stream()
            .map(Subnet::subnetId)
            .collect(Collectors.toList());

        CreateLoadBalancerRequest balancerRequest =
CreateLoadBalancerRequest.builder()
            .subnets(subnetIdStrings)
            .name(lbName)
            .scheme("internet-facing")
            .build();

        // Create and wait for the load balancer to become available.
        CreateLoadBalancerResponse lsResponse =
getLoadBalancerClient().createLoadBalancer(balancerRequest);
        String lbARN = lsResponse.loadBalancers().get(0).loadBalancerArn();

        ElasticLoadBalancingV2Waiter loadBalancerWaiter =
getLoadBalancerClient().waiter();
        DescribeLoadBalancersRequest request =
DescribeLoadBalancersRequest.builder()
            .loadBalancerArns(lbARN)
            .build();

        System.out.println("Waiting for Load Balancer " + lbName + " to become
available.");
        WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =
loadBalancerWaiter
            .waitUntilLoadBalancerAvailable(request);
        waiterResponse.matched().response().ifPresent(System.out::println);
        System.out.println("Load Balancer " + lbName + " is available.");

        // Get the DNS name (endpoint) of the load balancer.
        String lbDNSName = lsResponse.loadBalancers().get(0).dnsName();
        System.out.println("*** Load Balancer DNS Name: " + lbDNSName);

        // Create a listener for the load balance.
```

```
        Action action = Action.builder()
            .targetGroupArn(targetGroupARN)
            .type("forward")
            .build();

        CreateListenerRequest listenerRequest = CreateListenerRequest.builder()

.loadBalancerArn(lsResponse.loadBalancers().get(0).loadBalancerArn())
            .defaultActions(action)
            .port(port)
            .protocol(protocol)
            .defaultActions(action)
            .build();

        getLoadBalancerClient().createListener(listenerRequest);
        System.out.println("Created listener to forward traffic from load
balancer " + lbName + " to target group "
            + targetGroupARN);

        // Return the load balancer DNS name.
        return lbDNSName;

    } catch (ElasticLoadBalancingV2Exception e) {
        e.printStackTrace();
    }
    return "";
}
}
```

Crea una classe che utilizzi DynamoDB per simulare un servizio di raccomandazione.

```
public class Database {

    private static DynamoDbClient dynamoDbClient;

    public static DynamoDbClient getDynamoDbClient() {
        if (dynamoDbClient == null) {
            dynamoDbClient = DynamoDbClient.builder()
                .region(Region.US_EAST_1)
                .build();
        }
        return dynamoDbClient;
    }
}
```

```
}

// Checks to see if the Amazon DynamoDB table exists.
private boolean doesTableExist(String tableName) {
    try {
        // Describe the table and catch any exceptions.
        DescribeTableRequest describeTableRequest =
DescribeTableRequest.builder()
            .tableName(tableName)
            .build();

        getDynamoDbClient().describeTable(describeTableRequest);
        System.out.println("Table '" + tableName + "' exists.");
        return true;
    } catch (ResourceNotFoundException e) {
        System.out.println("Table '" + tableName + "' does not exist.");
    } catch (DynamoDbException e) {
        System.err.println("Error checking table existence: " + e.getMessage());
    }
    return false;
}

/**
 * Creates a DynamoDB table to use a recommendation service. The table has a
 * hash key named 'MediaType' that defines the type of media recommended, such
 * as
 * Book or Movie, and a range key named 'ItemId' that, combined with the
 * MediaType,
 * forms a unique identifier for the recommended item.
 */
public void createTable(String tableName, String fileName) throws IOException {
    // First check to see if the table exists.
    boolean doesExist = doesTableExist(tableName);
    if (!doesExist) {
        DynamoDbWaiter dbWaiter = getDynamoDbClient().waiter();
        CreateTableRequest createTableRequest = CreateTableRequest.builder()
            .tableName(tableName)
            .attributeDefinitions(
                AttributeDefinition.builder()
                    .attributeName("MediaType")
                    .attributeType(ScalarAttributeType.S)
                    .build(),
                AttributeDefinition.builder()

```

```

        .attributeName("ItemId")
        .attributeType(ScalarAttributeType.N)
        .build())
    .keySchema(
        KeySchemaElement.builder()
            .attributeName("MediaType")
            .keyType(KeyType.HASH)
            .build(),
        KeySchemaElement.builder()
            .attributeName("ItemId")
            .keyType(KeyType.RANGE)
            .build())
    .provisionedThroughput(
        ProvisionedThroughput.builder()
            .readCapacityUnits(5L)
            .writeCapacityUnits(5L)
            .build())
    .build();

    getDynamoDbClient().createTable(createTableRequest);
    System.out.println("Creating table " + tableName + "...");

    // Wait until the Amazon DynamoDB table is created.
    DescribeTableRequest tableRequest = DescribeTableRequest.builder()
        .tableName(tableName)
        .build();

    WaiterResponse<DescribeTableResponse> waiterResponse =
    dbWaiter.waitUntilTableExists(tableRequest);
    waiterResponse.matched().response().ifPresent(System.out::println);
    System.out.println("Table " + tableName + " created.");

    // Add records to the table.
    populateTable(fileName, tableName);
}

public void deleteTable(String tableName) {
    getDynamoDbClient().deleteTable(table -> table.tableName(tableName));
    System.out.println("Table " + tableName + " deleted.");
}

// Populates the table with data located in a JSON file using the DynamoDB
// enhanced client.

```

```

public void populateTable(String fileName, String tableName) throws IOException
{
    DynamoDbEnhancedClient enhancedClient = DynamoDbEnhancedClient.builder()
        .dynamoDbClient(getDynamoDbClient())
        .build();
    ObjectMapper objectMapper = new ObjectMapper();
    File jsonFile = new File(fileName);
    JsonNode rootNode = objectMapper.readTree(jsonFile);

    DynamoDbTable<Recommendation> mappedTable = enhancedClient.table(tableName,
        TableSchema.fromBean(Recommendation.class));
    for (JsonNode currentNode : rootNode) {
        String mediaType = currentNode.path("MediaType").path("S").asText();
        int itemId = currentNode.path("ItemId").path("N").asInt();
        String title = currentNode.path("Title").path("S").asText();
        String creator = currentNode.path("Creator").path("S").asText();

        // Create a Recommendation object and set its properties.
        Recommendation rec = new Recommendation();
        rec.setMediaType(mediaType);
        rec.setItemId(itemId);
        rec.setTitle(title);
        rec.setCreator(creator);

        // Put the item into the DynamoDB table.
        mappedTable.putItem(rec); // Add the Recommendation to the list.
    }
    System.out.println("Added all records to the " + tableName);
}
}

```

Crea una classe che racchiuda le operazioni di Systems Manager.

```

public class ParameterHelper {

    String tableName = "doc-example-resilient-architecture-table";
    String dyntable = "doc-example-recommendation-service";
    String failureResponse = "doc-example-resilient-architecture-failure-response";
    String healthCheck = "doc-example-resilient-architecture-health-check";

    public void reset() {
        put(dyntable, tableName);
    }
}

```

```
        put(failureResponse, "none");
        put(healthCheck, "shallow");
    }

    public void put(String name, String value) {
        SsmClient ssmClient = SsmClient.builder()
            .region(Region.US_EAST_1)
            .build();

        PutParameterRequest parameterRequest = PutParameterRequest.builder()
            .name(name)
            .value(value)
            .overwrite(true)
            .type("String")
            .build();

        ssmClient.putParameter(parameterRequest);
        System.out.printf("Setting demo parameter %s to '%s'.", name, value);
    }
}
```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella Documentazione di riferimento delle API AWS SDK for Java 2.x .
  - [AttachLoadBalancerTargetGroups](#)
  - [CreateAutoScalingGroup](#)
  - [CreateInstanceProfile](#)
  - [CreateLaunchTemplate](#)
  - [CreateListener](#)
  - [CreateLoadBalancer](#)
  - [CreateTargetGroup](#)
  - [DeleteAutoScalingGroup](#)
  - [DeleteInstanceProfile](#)
  - [DeleteLaunchTemplate](#)
  - [DeleteLoadBalancer](#)
  - [DeleteTargetGroup](#)
  - [DescribeAutoScalingGroups](#)




- [DescribeAvailabilityZones](#)
- [DescribeIamInstanceProfileAssociations](#)
- [DescribeInstances](#)
- [DescribeLoadBalancers](#)
- [DescribeSubnets](#)
- [DescribeTargetGroups](#)
- [DescribeTargetHealth](#)
- [DescribeVpcs](#)
- [RebootInstances](#)
- [ReplacesIamInstanceProfileAssociation](#)
- [TerminateInstanceInAutoScalingGroup](#)
- [UpdateAutoScalingGroup](#)

Gestisci gruppi e istanze

L'esempio di codice seguente mostra come:

- Crea un gruppo Amazon EC2 Auto Scaling con un modello di lancio e zone di disponibilità e ottieni informazioni sulle istanze in esecuzione.
- Abilita la raccolta di CloudWatch metriche Amazon.
- Aggiorna la capacità desiderata del gruppo e attendi l'avvio di un'istanza.
- Termina un'istanza nel gruppo.
- Elenca le attività di scalabilità che si verificano in risposta alle richieste degli utenti e ai cambiamenti di capacità.
- Ottieni statistiche per le CloudWatch metriche, quindi ripulisci le risorse.

SDK per Java 2.x

 Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
/**
 * Before running this SDK for Java (v2) code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * In addition, create a launch template. For more information, see the
 * following topic:
 *
 * https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-launch-
templates.html#create-launch-template
 *
 * This code example performs the following operations:
 * 1. Creates an Auto Scaling group using an AutoScalingWaiter.
 * 2. Gets a specific Auto Scaling group and returns an instance Id value.
 * 3. Describes Auto Scaling with the Id value.
 * 4. Enables metrics collection.
 * 5. Update an Auto Scaling group.
 * 6. Describes Account details.
 * 7. Describe account details"
 * 8. Updates an Auto Scaling group to use an additional instance.
 * 9. Gets the specific Auto Scaling group and gets the number of instances.
 * 10. List the scaling activities that have occurred for the group.
 * 11. Terminates an instance in the Auto Scaling group.
 * 12. Stops the metrics collection.
 * 13. Deletes the Auto Scaling group.
 */

public class AutoScalingScenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    public static void main(String[] args) throws InterruptedException {
        final String usage = ""

            Usage:
                <groupName> <launchTemplateName> <vpcZoneId>

            Where:
                groupName - The name of the Auto Scaling group.
                launchTemplateName - The name of the launch template.\s
    }
}
```

```
        vpcZoneId - A subnet Id for a virtual private cloud (VPC) where
instances in the Auto Scaling group can be created.
        """;

    if (args.length != 3) {
        System.out.println(usage);
        System.exit(1);
    }

    String groupName = args[0];
    String launchTemplateName = args[1];
    String vpcZoneId = args[2];
    AutoScalingClient autoScalingClient = AutoScalingClient.builder()
        .region(Region.US_EAST_1)
        .build();

    System.out.println(DASHES);
    System.out.println("Welcome to the Amazon EC2 Auto Scaling example
scenario.");
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("1. Create an Auto Scaling group named " + groupName);
    createAutoScalingGroup(autoScalingClient, groupName, launchTemplateName,
vpcZoneId);
    System.out.println(
        "Wait 1 min for the resources, including the instance. Otherwise, an
empty instance Id is returned");
    Thread.sleep(60000);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("2. Get Auto Scale group Id value");
    String instanceId = getSpecificAutoScalingGroups(autoScalingClient,
groupName);
    if (instanceId.compareTo("") == 0) {
        System.out.println("Error - no instance Id value");
        System.exit(1);
    } else {
        System.out.println("The instance Id value is " + instanceId);
    }
    System.out.println(DASHES);

    System.out.println(DASHES);
```

```
        System.out.println("3. Describe Auto Scaling with the Id value " +
instanceId);
        describeAutoScalingInstance(autoScalingClient, instanceId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. Enable metrics collection " + instanceId);
        enableMetricsCollection(autoScalingClient, groupName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. Update an Auto Scaling group to update max size to
3");
        updateAutoScalingGroup(autoScalingClient, groupName, launchTemplateName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6. Describe Auto Scaling groups");
        describeAutoScalingGroups(autoScalingClient, groupName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Describe account details");
        describeAccountLimits(autoScalingClient);
        System.out.println(
            "Wait 1 min for the resources, including the instance. Otherwise, an
empty instance Id is returned");
        Thread.sleep(60000);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("8. Set desired capacity to 2");
        setDesiredCapacity(autoScalingClient, groupName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("9. Get the two instance Id values and state");
        getSpecificAutoScalingGroups(autoScalingClient, groupName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("10. List the scaling activities that have occurred for
the group");
        describeScalingActivities(autoScalingClient, groupName);
```

```
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("11. Terminate an instance in the Auto Scaling group");
        terminateInstanceInAutoScalingGroup(autoScalingClient, instanceId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("12. Stop the metrics collection");
        disableMetricsCollection(autoScalingClient, groupName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("13. Delete the Auto Scaling group");
        deleteAutoScalingGroup(autoScalingClient, groupName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("The Scenario has successfully completed.");
        System.out.println(DASHES);

        autoScalingClient.close();
    }

    public static void describeScalingActivities(AutoScalingClient
autoScalingClient, String groupName) {
        try {
            DescribeScalingActivitiesRequest scalingActivitiesRequest =
DescribeScalingActivitiesRequest.builder()
                .autoScalingGroupName(groupName)
                .maxRecords(10)
                .build();

            DescribeScalingActivitiesResponse response = autoScalingClient
                .describeScalingActivities(scalingActivitiesRequest);
            List<Activity> activities = response.activities();
            for (Activity activity : activities) {
                System.out.println("The activity Id is " + activity.activityId());
                System.out.println("The activity details are " +
activity.details());
            }

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }
    }
}
```

```
        System.exit(1);
    }
}

public static void setDesiredCapacity(AutoScalingClient autoScalingClient,
String groupName) {
    try {
        SetDesiredCapacityRequest capacityRequest =
SetDesiredCapacityRequest.builder()
            .autoScalingGroupName(groupName)
            .desiredCapacity(2)
            .build();

        autoScalingClient.setDesiredCapacity(capacityRequest);
        System.out.println("You have set the DesiredCapacity to 2");

    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void createAutoScalingGroup(AutoScalingClient autoScalingClient,
String groupName,
String launchTemplateName,
String vpcZoneId) {
    try {
        AutoScalingWaiter waiter = autoScalingClient.waiter();
        LaunchTemplateSpecification templateSpecification =
LaunchTemplateSpecification.builder()
            .launchTemplateName(launchTemplateName)
            .build();

        CreateAutoScalingGroupRequest request =
CreateAutoScalingGroupRequest.builder()
            .autoScalingGroupName(groupName)
            .availabilityZones("us-east-1a")
            .launchTemplate(templateSpecification)
            .maxSize(1)
            .minSize(1)
            .vpcZoneIdentifier(vpcZoneId)
            .build();

        autoScalingClient.createAutoScalingGroup(request);
    }
}
```

```
        DescribeAutoScalingGroupsRequest groupsRequest =
DescribeAutoScalingGroupsRequest.builder()
        .autoScalingGroupNames(groupName)
        .build();

        WaiterResponse<DescribeAutoScalingGroupsResponse> waiterResponse =
waiter
        .waitUntilGroupExists(groupsRequest);
        waiterResponse.matched().response().ifPresent(System.out::println);
        System.out.println("Auto Scaling Group created");

    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void describeAutoScalingInstance(AutoScalingClient
autoScalingClient, String id) {
    try {
        DescribeAutoScalingInstancesRequest describeAutoScalingInstancesRequest
= DescribeAutoScalingInstancesRequest
        .builder()
        .instanceIds(id)
        .build();

        DescribeAutoScalingInstancesResponse response = autoScalingClient
        .describeAutoScalingInstances(describeAutoScalingInstancesRequest);
        List<AutoScalingInstanceDetails> instances =
response.autoScalingInstances();
        for (AutoScalingInstanceDetails instance : instances) {
            System.out.println("The instance lifecycle state is: " +
instance.lifecycleState());
        }

    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void describeAutoScalingGroups(AutoScalingClient
autoScalingClient, String groupName) {
```

```
    try {
        DescribeAutoScalingGroupsRequest groupsRequest =
DescribeAutoScalingGroupsRequest.builder()
            .autoScalingGroupNames(groupName)
            .maxRecords(10)
            .build();

        DescribeAutoScalingGroupsResponse response =
autoScalingClient.describeAutoScalingGroups(groupsRequest);
        List<AutoScalingGroup> groups = response.autoScalingGroups();
        for (AutoScalingGroup group : groups) {
            System.out.println("*** The service to use for the health checks: "
+ group.healthCheckType());
        }

    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static String getSpecificAutoScalingGroups(AutoScalingClient
autoScalingClient, String groupName) {
    try {
        String instanceId = "";
        DescribeAutoScalingGroupsRequest scalingGroupsRequest =
DescribeAutoScalingGroupsRequest.builder()
            .autoScalingGroupNames(groupName)
            .build();

        DescribeAutoScalingGroupsResponse response = autoScalingClient
            .describeAutoScalingGroups(scalingGroupsRequest);
        List<AutoScalingGroup> groups = response.autoScalingGroups();
        for (AutoScalingGroup group : groups) {
            System.out.println("The group name is " +
group.autoScalingGroupName());
            System.out.println("The group ARN is " +
group.autoScalingGroupARN());
            List<Instance> instances = group.instances();

            for (Instance instance : instances) {
                instanceId = instance.instanceId();
                System.out.println("The instance id is " + instanceId);
            }
        }
    }
}
```



```
        System.out.println("The lifecycle state is " +
instance.lifecycleState());
    }
}

    return instanceId;
} catch (AutoScalingException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
return "";
}

public static void enableMetricsCollection(AutoScalingClient autoScalingClient,
String groupName) {
    try {
        EnableMetricsCollectionRequest collectionRequest =
EnableMetricsCollectionRequest.builder()
            .autoScalingGroupName(groupName)
            .metrics("GroupMaxSize")
            .granularity("1Minute")
            .build();

        autoScalingClient.enableMetricsCollection(collectionRequest);
        System.out.println("The enable metrics collection operation was
successful");

    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void disableMetricsCollection(AutoScalingClient autoScalingClient,
String groupName) {
    try {
        DisableMetricsCollectionRequest disableMetricsCollectionRequest =
DisableMetricsCollectionRequest.builder()
            .autoScalingGroupName(groupName)
            .metrics("GroupMaxSize")
            .build();

        autoScalingClient.disableMetricsCollection(disableMetricsCollectionRequest);
```

```
        System.out.println("The disable metrics collection operation was
successful");

    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void describeAccountLimits(AutoScalingClient autoScalingClient) {
    try {
        DescribeAccountLimitsResponse response =
autoScalingClient.describeAccountLimits();
        System.out.println("The max number of auto scaling groups is " +
response.maxNumberOfAutoScalingGroups());
        System.out.println("The current number of auto scaling groups is " +
response.numberOfWorkingAutoScalingGroups());

    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void updateAutoScalingGroup(AutoScalingClient autoScalingClient,
String groupName,
    String launchTemplateName) {
    try {
        AutoScalingWaiter waiter = autoScalingClient.waiter();
        LaunchTemplateSpecification templateSpecification =
LaunchTemplateSpecification.builder()
            .launchTemplateName(launchTemplateName)
            .build();

        UpdateAutoScalingGroupRequest groupRequest =
UpdateAutoScalingGroupRequest.builder()
            .maxSize(3)
            .autoScalingGroupName(groupName)
            .launchTemplate(templateSpecification)
            .build();

        autoScalingClient.updateAutoScalingGroup(groupRequest);
        DescribeAutoScalingGroupsRequest groupsRequest =
DescribeAutoScalingGroupsRequest.builder()
```

```
        .autoScalingGroupNames(groupName)
        .build();

    WaiterResponse<DescribeAutoScalingGroupsResponse> waiterResponse =
waiter
        .waitUntilGroupInService(groupsRequest);
    waiterResponse.matched().response().ifPresent(System.out::println);
    System.out.println("You successfully updated the auto scaling group " +
groupName);

    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

    public static void terminateInstanceInAutoScalingGroup(AutoScalingClient
autoScalingClient, String instanceId) {
        try {
            TerminateInstanceInAutoScalingGroupRequest request =
TerminateInstanceInAutoScalingGroupRequest.builder()
                .instanceId(instanceId)
                .shouldDecrementDesiredCapacity(false)
                .build();

            autoScalingClient.terminateInstanceInAutoScalingGroup(request);
            System.out.println("You have terminated instance " + instanceId);

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void deleteAutoScalingGroup(AutoScalingClient autoScalingClient,
String groupName) {
        try {
            DeleteAutoScalingGroupRequest deleteAutoScalingGroupRequest =
DeleteAutoScalingGroupRequest.builder()
                .autoScalingGroupName(groupName)
                .forceDelete(true)
                .build();

            autoScalingClient.deleteAutoScalingGroup(deleteAutoScalingGroupRequest);
```

```
        System.out.println("You successfully deleted " + groupName);

    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella Documentazione di riferimento delle API AWS SDK for Java 2.x .
  - [CreateAutoScalingGroup](#)
  - [DeleteAutoScalingGroup](#)
  - [DescribeAutoScalingGroups](#)
  - [DescribeAutoScalingInstances](#)
  - [DescribeScalingActivities](#)
  - [DisableMetricsCollection](#)
  - [EnableMetricsCollection](#)
  - [SetDesiredCapacity](#)
  - [TerminateInstanceInAutoScalingGroup](#)
  - [UpdateAutoScalingGroup](#)

## Esempi di Amazon Bedrock con SDK for Java 2.x

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando Amazon Bedrock. AWS SDK for Java 2.x

Le operazioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le operazioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati e negli esempi tra servizi.

Scenari: esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio.

Ogni esempio include un collegamento a GitHub, dove puoi trovare istruzioni su come configurare ed eseguire il codice nel contesto.

## Argomenti

- [Azioni](#)

## Azioni

### GetFoundationModel

Il seguente esempio di codice mostra come utilizzare `GetFoundationModel`.

#### SDK per Java 2.x

#### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Ottieni dettagli su un modello base utilizzando il client Amazon Bedrock sincrono.

```
/**
 * Get details about an Amazon Bedrock foundation model.
 *
 * @param bedrockClient The service client for accessing Amazon Bedrock.
 * @param modelIdentifier The model identifier.
 * @return An object containing the foundation model's details.
 */
public static FoundationModelDetails getFoundationModel(BedrockClient
bedrockClient, String modelIdentifier) {
    try {
        GetFoundationModelResponse response = bedrockClient.getFoundationModel(
            r -> r.modelIdentifier(modelIdentifier)
        );

        FoundationModelDetails model = response.modelDetails();

        System.out.println(" Model ID:                " + model.modelId());
        System.out.println(" Model ARN:                " +
model.modelArn());
        System.out.println(" Model Name:                " +
model.modelName());
        System.out.println(" Provider Name:            " +
model.providerName());
    }
}
```

```

        System.out.println(" Lifecycle status:           " +
model.modelLifecycle().statusAsString());
        System.out.println(" Input modalities:           " +
model.inputModalities());
        System.out.println(" Output modalities:           " +
model.outputModalities());
        System.out.println(" Supported customizations:       " +
model.customizationsSupported());
        System.out.println(" Supported inference types:      " +
model.inferenceTypesSupported());
        System.out.println(" Response streaming supported:  " +
model.responseStreamingSupported());

        return model;

    } catch (ValidationException e) {
        throw new IllegalArgumentException(e.getMessage());
    } catch (SdkException e) {
        System.err.println(e.getMessage());
        throw new RuntimeException(e);
    }
}

```

Ottieni dettagli su un modello base che utilizza il client asincrono Amazon Bedrock.

```

/**
 * Get details about an Amazon Bedrock foundation model.
 *
 * @param bedrockClient The async service client for accessing Amazon Bedrock.
 * @param modelIdentifier The model identifier.
 * @return An object containing the foundation model's details.
 */
public static FoundationModelDetails getFoundationModel(BedrockAsyncClient
bedrockClient, String modelIdentifier) {
    try {
        CompletableFuture<GetFoundationModelResponse> future =
bedrockClient.getFoundationModel(
            r -> r.modelIdentifier(modelIdentifier)
        );

        FoundationModelDetails model = future.get().modelDetails();
    }
}

```

```

        System.out.println(" Model ID:                " + model.modelId());
        System.out.println(" Model ARN:                " +
model.modelArn());
        System.out.println(" Model Name:                " +
model.modelName());
        System.out.println(" Provider Name:            " +
model.providerName());
        System.out.println(" Lifecycle status:        " +
model.modelLifecycle().statusAsString());
        System.out.println(" Input modalities:        " +
model.inputModalities());
        System.out.println(" Output modalities:       " +
model.outputModalities());
        System.out.println(" Supported customizations: " +
model.customizationsSupported());
        System.out.println(" Supported inference types: " +
model.inferenceTypesSupported());
        System.out.println(" Response streaming supported: " +
model.responseStreamingSupported());

        return model;

    } catch (ExecutionException e) {
        if (e.getMessage().contains("ValidationException")) {
            throw new IllegalArgumentException(e.getMessage());
        } else {
            System.err.println(e.getMessage());
            throw new RuntimeException(e);
        }
    } catch (InterruptedException e) {
        Thread.currentThread().interrupt();
        System.err.println(e.getMessage());
        throw new RuntimeException(e);
    }
}

```

- Per i dettagli sulle API, consulta [GetFoundationModel](#) nella sezione API Reference.AWS SDK for Java 2.x

## ListFoundationModels

Il seguente esempio di codice mostra come utilizzare `ListFoundationModels`.

## SDK per Java 2.x

 Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Elenca i modelli Amazon Bedrock Foundation disponibili utilizzando il client Amazon Bedrock sincrono.

```
/**
 * Lists Amazon Bedrock foundation models that you can use.
 * You can filter the results with the request parameters.
 *
 * @param bedrockClient The service client for accessing Amazon Bedrock.
 * @return A list of objects containing the foundation models' details
 */
public static List<FoundationModelSummary> listFoundationModels(BedrockClient
bedrockClient) {

    try {
        ListFoundationModelsResponse response =
bedrockClient.listFoundationModels(r -> {});

        List<FoundationModelSummary> models = response.modelSummaries();

        if (models.isEmpty()) {
            System.out.println("No available foundation models in " +
region.toString());
        } else {
            for (FoundationModelSummary model : models) {
                System.out.println("Model ID: " + model.modelId());
                System.out.println("Provider: " + model.providerName());
                System.out.println("Name:      " + model.modelName());
                System.out.println();
            }
        }

        return models;

    } catch (SdkClientException e) {
```



```

        System.err.println(e.getMessage());
        throw new RuntimeException(e);
    }
}

```

Elenca i modelli Amazon Bedrock Foundation disponibili utilizzando il client Amazon Bedrock asincrono.

```

/**
 * Lists Amazon Bedrock foundation models that you can use.
 * You can filter the results with the request parameters.
 *
 * @param bedrockClient The async service client for accessing Amazon Bedrock.
 * @return A list of objects containing the foundation models' details
 */
public static List<FoundationModelSummary>
listFoundationModels(BedrockAsyncClient bedrockClient) {
    try {
        CompletableFuture<ListFoundationModelsResponse> future =
bedrockClient.listFoundationModels(r -> {});

        List<FoundationModelSummary> models = future.get().modelSummaries();

        if (models.isEmpty()) {
            System.out.println("No available foundation models in " +
region.toString());
        } else {
            for (FoundationModelSummary model : models) {
                System.out.println("Model ID: " + model.modelId());
                System.out.println("Provider: " + model.providerName());
                System.out.println("Name:      " + model.modelName());
                System.out.println();
            }
        }

        return models;
    } catch (InterruptedException e) {
        Thread.currentThread().interrupt();
        System.err.println(e.getMessage());
        throw new RuntimeException(e);
    } catch (ExecutionException e) {

```

```
        System.err.println(e.getMessage());
        throw new RuntimeException(e);
    }
}
```

- Per i dettagli sulle API, consulta la sezione API Reference. [ListFoundationModels](#) AWS SDK for Java 2.x

## Esempi di Amazon Bedrock Runtime con SDK for Java 2.x

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando AWS SDK for Java 2.x with Amazon Bedrock Runtime.

Le operazioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le operazioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati e negli esempi tra servizi.

Scenari: esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio.

Ogni esempio include un collegamento a GitHub, dove puoi trovare istruzioni su come configurare ed eseguire il codice nel contesto.

### Argomenti

- [AI21 Labs Jurassic-2](#)
- [Amazon Titan Image Generator](#)
- [Testo Amazon Titan](#)
- [Amazon Titan Text Embeddings](#)
- [Anthropic Claude](#)
- [Cohere Command](#)
- [Meta Llama](#)
- [IA Mistral](#)
- [Scenari](#)
- [Diffusione stabile](#)

## AI21 Labs Jurassic-2

conversare

Il seguente esempio di codice mostra come inviare un messaggio di testo a AI21 Labs Jurassic-2, utilizzando l'API Converse di Bedrock.

SDK per Java 2.x

### Note

C'è altro su [GitHub](#) Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Invia un messaggio di testo a AI21 Labs Jurassic-2, utilizzando l'API Converse di Bedrock.

```
// Use the Converse API to send a text message to AI21 Labs Jurassic-2.

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.exception.SdkClientException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeClient;
import software.amazon.awssdk.services.bedrockruntime.model.ContentBlock;
import software.amazon.awssdk.services.bedrockruntime.model.ConversationRole;
import software.amazon.awssdk.services.bedrockruntime.model.ConverseResponse;
import software.amazon.awssdk.services.bedrockruntime.model.Message;

public class Converse {

    public static String converse() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
        provider.
        var client = BedrockRuntimeClient.builder()
            .credentialsProvider(DefaultCredentialsProvider.create())
            .region(Region.US_EAST_1)
            .build();

        // Set the model ID, e.g., Jurassic-2 Mid.
        var modelId = "ai21.j2-mid-v1";
```

```
        // Create the input text and embed it in a message object with the user
        role.
        var inputText = "Describe the purpose of a 'hello world' program in one
        line.";
        var message = Message.builder()
            .content(ContentBlock.fromText(inputText))
            .role(ConversationRole.USER)
            .build();

        try {
            // Send the message with a basic inference configuration.
            ConverseResponse response = client.converse(request -> request
                .modelId(modelId)
                .messages(message)
                .inferenceConfig(config -> config
                    .maxTokens(512)
                    .temperature(0.5F)
                    .topP(0.9F)));

            // Retrieve the generated text from Bedrock's response object.
            var responseText = response.output().message().content().get(0).text();
            System.out.println(responseText);

            return responseText;

        } catch (SdkClientException e) {
            System.err.printf("ERROR: Can't invoke '%s'. Reason: %s", modelId,
            e.getMessage());
            throw new RuntimeException(e);
        }
    }

    public static void main(String[] args) {
        converse();
    }
}
```

Invia un messaggio di testo a AI21 Labs Jurassic-2, utilizzando l'API Converse di Bedrock con il client Java asincrono.

```
// Use the Converse API to send a text message to AI21 Labs Jurassic-2
```

```
// with the async Java client.

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
import software.amazon.awssdk.services.bedrockruntime.model.ContentBlock;
import software.amazon.awssdk.services.bedrockruntime.model.ConversationRole;
import software.amazon.awssdk.services.bedrockruntime.model.Message;

import java.util.concurrent.CompletableFuture;
import java.util.concurrent.ExecutionException;

public class ConverseAsync {

    public static String converseAsync() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
        provider.
        var client = BedrockRuntimeAsyncClient.builder()
            .credentialsProvider(DefaultCredentialsProvider.create())
            .region(Region.US_EAST_1)
            .build();

        // Set the model ID, e.g., Jurassic-2 Mid.
        var modelId = "ai21.j2-mid-v1";

        // Create the input text and embed it in a message object with the user
        role.
        var inputText = "Describe the purpose of a 'hello world' program in one
        line.";
        var message = Message.builder()
            .content(ContentBlock.fromText(inputText))
            .role(ConversationRole.USER)
            .build();

        // Send the message with a basic inference configuration.
        var request = client.converse(params -> params
            .modelId(modelId)
            .messages(message)
            .inferenceConfig(config -> config
                .maxTokens(512)
                .temperature(0.5F)
                .topP(0.9F))
```

```
);

// Prepare a future object to handle the asynchronous response.
CompletableFuture<String> future = new CompletableFuture<>();

// Handle the response or error using the future object.
request.whenComplete((response, error) -> {
    if (error == null) {
        // Extract the generated text from Bedrock's response object.
        String responseText =
response.output().message().content().get(0).text();
        future.complete(responseText);
    } else {
        future.completeExceptionally(error);
    }
});

try {
    // Wait for the future object to complete and retrieve the generated
text.
    String responseText = future.get();
    System.out.println(responseText);

    return responseText;

} catch (ExecutionException | InterruptedException e) {
    System.err.printf("Can't invoke '%s': %s", modelId, e.getMessage());
    throw new RuntimeException(e);
}

}

public static void main(String[] args) {
    converseAsync();
}
}
```

- [Per i dettagli sulle API, consulta \*Converse in API Reference.AWS SDK for Java 2.x\*](#)

## InvokeModel

Il seguente esempio di codice mostra come inviare un messaggio di testo a AI21 Labs Jurassic-2, utilizzando l'API Invoke Model.

## SDK per Java 2.x

 Note

C'è altro su [GitHub](#) Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Usa l'API Invoke Model per inviare un messaggio di testo.

```
// Use the native inference API to send a text message to AI21 Labs Jurassic-2.

import org.json.JSONObject;
import org.json.JSONPointer;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.core.exception.SdkClientException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeClient;

public class InvokeModel {

    public static String invokeModel() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
        provider.
        var client = BedrockRuntimeClient.builder()
            .credentialsProvider(DefaultCredentialsProvider.create())
            .region(Region.US_EAST_1)
            .build();

        // Set the model ID, e.g., Jurassic-2 Mid.
        var modelId = "ai21.j2-mid-v1";

        // The InvokeModel API uses the model's native payload.
        // Learn more about the available inference parameters and response fields
        at:
        // https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-
        jurassic2.html
        var nativeRequestTemplate = "{ \"prompt\": \"{{prompt}}\" }";

        // Define the prompt for the model.
```

```
var prompt = "Describe the purpose of a 'hello world' program in one line.";

// Embed the prompt in the model's native request payload.
String nativeRequest = nativeRequestTemplate.replace("{{prompt}}", prompt);

try {
    // Encode and send the request to the Bedrock Runtime.
    var response = client.invokeModel(request -> request
        .body(SdkBytes.fromUtf8String(nativeRequest))
        .modelId(modelId)
    );

    // Decode the response body.
    var responseBody = new JSONObject(response.body().asUtf8String());

    // Retrieve the generated text from the model's response.
    var text = new JSONPointer("/completions/0/data/text").queryFrom(responseBody).toString();
    System.out.println(text);

    return text;

} catch (SdkClientException e) {
    System.err.printf("ERROR: Can't invoke '%s'. Reason: %s", modelId,
e.getMessage());
    throw new RuntimeException(e);
}

public static void main(String[] args) {
    invokeModel();
}
}
```

- Per i dettagli sull'API, consulta [InvokeModel](#) in AWS SDK for Java 2.x API Reference.

## Amazon Titan Image Generator

### InvokeModel

Il seguente esempio di codice mostra come richiamare Amazon Titan Image su Amazon Bedrock per generare un'immagine.



## SDK per Java 2.x

 Note

C'è di più su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Crea un'immagine con Amazon Titan Image Generator.

```
// Create an image with the Amazon Titan Image Generator.

import org.json.JSONObject;
import org.json.JSONPointer;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.core.exception.SdkClientException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeClient;

import java.math.BigInteger;
import java.security.SecureRandom;

import static com.example.bedrockruntime.libs.ImageTools.displayImage;

public class InvokeModel {

    public static String invokeModel() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
        provider.
        var client = BedrockRuntimeClient.builder()
            .credentialsProvider(DefaultCredentialsProvider.create())
            .region(Region.US_EAST_1)
            .build();

        // Set the model ID, e.g., Titan Image G1.
        var modelId = "amazon.titan-image-generator-v1";

        // The InvokeModel API uses the model's native payload.
        // Learn more about the available inference parameters and response fields
        at:
```

```
// https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-
titan-image.html
var nativeRequestTemplate = """
    {
        "taskType": "TEXT_IMAGE",
        "textToImageParams": { "text": "{{prompt}}" },
        "imageGenerationConfig": { "seed": {{seed}} }
    }""";

// Define the prompt for the image generation.
var prompt = "A stylized picture of a cute old steampunk robot";

// Get a random 31-bit seed for the image generation (max. 2,147,483,647).
var seed = new BigInteger(31, new SecureRandom());

// Embed the prompt and seed in the model's native request payload.
var nativeRequest = nativeRequestTemplate
    .replace("{{prompt}}", prompt)
    .replace("{{seed}}", seed.toString());

try {
    // Encode and send the request to the Bedrock Runtime.
    var response = client.invokeModel(request -> request
        .body(SdkBytes.fromUtf8String(nativeRequest))
        .modelId(modelId)
    );

    // Decode the response body.
    var responseBody = new JSONObject(response.body().asUtf8String());

    // Retrieve the generated image data from the model's response.
    var base64ImageData = new JSONObject("/
images/0").queryFrom(responseBody).toString();

    return base64ImageData;

} catch (SdkClientException e) {
    System.err.printf("ERROR: Can't invoke '%s'. Reason: %s", modelId,
e.getMessage());
    throw new RuntimeException(e);
}
}

public static void main(String[] args) {
```

```
        System.out.println("Generating image. This may take a few seconds...");

        String base64ImageData = invokeModel();

        displayImage(base64ImageData);
    }
}
```

- Per i dettagli sull'API, consulta la sezione [InvokeModel AWS SDK for Java 2.x API Reference](#).

## Testo Amazon Titan

### conversare

Il seguente esempio di codice mostra come inviare un messaggio di testo ad Amazon Titan Text, utilizzando l'API Converse di Bedrock.

### SDK per Java 2.x

#### Note

C'è di più su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Invia un messaggio di testo ad Amazon Titan Text utilizzando l'API Converse di Bedrock.

```
// Use the Converse API to send a text message to Amazon Titan Text.

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.exception.SdkClientException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeClient;
import software.amazon.awssdk.services.bedrockruntime.model.ContentBlock;
import software.amazon.awssdk.services.bedrockruntime.model.ConversationRole;
import software.amazon.awssdk.services.bedrockruntime.model.ConverseResponse;
import software.amazon.awssdk.services.bedrockruntime.model.Message;

public class Converse {

    public static String converse() {
```

```
// Create a Bedrock Runtime client in the AWS Region you want to use.
// Replace the DefaultCredentialsProvider with your preferred credentials
provider.
var client = BedrockRuntimeClient.builder()
    .credentialsProvider(DefaultCredentialsProvider.create())
    .region(Region.US_EAST_1)
    .build();

// Set the model ID, e.g., Titan Text Premier.
var modelId = "amazon.titan-text-premier-v1:0";

// Create the input text and embed it in a message object with the user
role.
var inputText = "Describe the purpose of a 'hello world' program in one
line.";
var message = Message.builder()
    .content(ContentBlock.fromText(inputText))
    .role(ConversationRole.USER)
    .build();

try {
    // Send the message with a basic inference configuration.
    ConverseResponse response = client.converse(request -> request
        .modelId(modelId)
        .messages(message)
        .inferenceConfig(config -> config
            .maxTokens(512)
            .temperature(0.5F)
            .topP(0.9F)));

    // Retrieve the generated text from Bedrock's response object.
    var responseText = response.output().message().content().get(0).text();
    System.out.println(responseText);

    return responseText;

} catch (SdkClientException e) {
    System.err.printf("ERROR: Can't invoke '%s'. Reason: %s", modelId,
e.getMessage());
    throw new RuntimeException(e);
}
}
```

```
    public static void main(String[] args) {
        converse();
    }
}
```

Invia un messaggio di testo ad Amazon Titan Text utilizzando l'API Converse di Bedrock con il client Java asincrono.

```
// Use the Converse API to send a text message to Amazon Titan Text
// with the async Java client.

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
import software.amazon.awssdk.services.bedrockruntime.model.ContentBlock;
import software.amazon.awssdk.services.bedrockruntime.model.ConversationRole;
import software.amazon.awssdk.services.bedrockruntime.model.Message;

import java.util.concurrent.CompletableFuture;
import java.util.concurrent.ExecutionException;

public class ConverseAsync {

    public static String converseAsync() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
        provider.
        var client = BedrockRuntimeAsyncClient.builder()
            .credentialsProvider(DefaultCredentialsProvider.create())
            .region(Region.US_EAST_1)
            .build();

        // Set the model ID, e.g., Titan Text Premier.
        var modelId = "amazon.titan-text-premier-v1:0";

        // Create the input text and embed it in a message object with the user
        role.
        var inputText = "Describe the purpose of a 'hello world' program in one
        line.";
```

```
var message = Message.builder()
    .content(ContentBlock.fromText(inputText))
    .role(ConversationRole.USER)
    .build();

// Send the message with a basic inference configuration.
var request = client.converse(params -> params
    .modelId(modelId)
    .messages(message)
    .inferenceConfig(config -> config
        .maxTokens(512)
        .temperature(0.5F)
        .topP(0.9F))
);

// Prepare a future object to handle the asynchronous response.
CompletableFuture<String> future = new CompletableFuture<>();

// Handle the response or error using the future object.
request.whenComplete((response, error) -> {
    if (error == null) {
        // Extract the generated text from Bedrock's response object.
        String responseText =
response.output().message().content().get(0).text();
        future.complete(responseText);
    } else {
        future.completeExceptionally(error);
    }
});

try {
    // Wait for the future object to complete and retrieve the generated
text.
    String responseText = future.get();
    System.out.println(responseText);

    return responseText;
} catch (ExecutionException | InterruptedException e) {
    System.err.printf("Can't invoke '%s': %s", modelId, e.getMessage());
    throw new RuntimeException(e);
}
}
```

```
    public static void main(String[] args) {  
        converseAsync();  
    }  
}
```

- [Per i dettagli sull'API, consulta Converse in API Reference.AWS SDK for Java 2.x](#)

## ConverseStream

Il seguente esempio di codice mostra come inviare un messaggio di testo ad Amazon Titan Text, utilizzando l'API Converse di Bedrock ed elaborare il flusso di risposta in tempo reale.

### SDK per Java 2.x

#### Note

C'è altro su [GitHub](#) Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Invia un messaggio di testo ad Amazon Titan Text, utilizzando l'API Converse di Bedrock ed elabora il flusso di risposta in tempo reale.

```
// Use the Converse API to send a text message to Amazon Titan Text  
// and print the response stream.  
  
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;  
import software.amazon.awssdk.regions.Region;  
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;  
import software.amazon.awssdk.services.bedrockruntime.model.ContentBlock;  
import software.amazon.awssdk.services.bedrockruntime.model.ConversationRole;  
import  
    software.amazon.awssdk.services.bedrockruntime.model.ConverseStreamResponseHandler;  
import software.amazon.awssdk.services.bedrockruntime.model.Message;  
  
import java.util.concurrent.ExecutionException;  
  
public class ConverseStream {  
  
    public static void main(String[] args) {
```

```
// Create a Bedrock Runtime client in the AWS Region you want to use.
// Replace the DefaultCredentialsProvider with your preferred credentials
provider.
var client = BedrockRuntimeAsyncClient.builder()
    .credentialsProvider(DefaultCredentialsProvider.create())
    .region(Region.US_EAST_1)
    .build();

// Set the model ID, e.g., Titan Text Premier.
var modelId = "amazon.titan-text-premier-v1:0";

// Create the input text and embed it in a message object with the user
role.
var inputText = "Describe the purpose of a 'hello world' program in one
line.";
var message = Message.builder()
    .content(ContentBlock.fromText(inputText))
    .role(ConversationRole.USER)
    .build();

// Create a handler to extract and print the response text in real-time.
var responseStreamHandler = ConverseStreamResponseHandler.builder()
    .subscriber(ConverseStreamResponseHandler.Visitor.builder()
        .onContentBlockDelta(chunk -> {
            String responseText = chunk.delta().text();
            System.out.print(responseText);
        }).build()
    ).onError(err ->
        System.err.printf("Can't invoke '%s': %s", modelId,
err.getMessage())
    ).build();

try {
    // Send the message with a basic inference configuration and attach the
handler.
    client.converseStream(request -> request
        .modelId(modelId)
        .messages(message)
        .inferenceConfig(config -> config
            .maxTokens(512)
            .temperature(0.5F)
            .topP(0.9F)
        ), responseStreamHandler).get();
```



```
        } catch (ExecutionException | InterruptedException e) {
            System.err.printf("Can't invoke '%s': %s", modelId,
                e.getCause().getMessage());
        }
    }
}
```

- Per i dettagli sull'API, consulta API [ConverseStream](#) Reference AWS SDK for Java 2.x .

## InvokeModel

Il seguente esempio di codice mostra come inviare un messaggio di testo ad Amazon Titan Text utilizzando l'API Invoke Model.

### SDK per Java 2.x

#### Note

C'è di più su. [GitHub](#) Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Usa l'API Invoke Model per inviare un messaggio di testo.

```
// Use the native inference API to send a text message to Amazon Titan Text.

import org.json.JSONObject;
import org.json.JSONPointer;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.core.exception.SdkClientException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeClient;

public class InvokeModel {

    public static String invokeModel() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
        provider.
```

```
var client = BedrockRuntimeClient.builder()
    .credentialsProvider(DefaultCredentialsProvider.create())
    .region(Region.US_EAST_1)
    .build();

// Set the model ID, e.g., Titan Text Premier.
var modelId = "amazon.titan-text-premier-v1:0";

// The InvokeModel API uses the model's native payload.
// Learn more about the available inference parameters and response fields
at:
// https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-
titan-text.html
var nativeRequestTemplate = "{ \"inputText\": \"{{prompt}}\" }";

// Define the prompt for the model.
var prompt = "Describe the purpose of a 'hello world' program in one line.";

// Embed the prompt in the model's native request payload.
String nativeRequest = nativeRequestTemplate.replace("{{prompt}}", prompt);

try {
    // Encode and send the request to the Bedrock Runtime.
    var response = client.invokeModel(request -> request
        .body(SdkBytes.fromUtf8String(nativeRequest))
        .modelId(modelId)
    );

    // Decode the response body.
    var responseBody = new JSONObject(response.body().asUtf8String());

    // Retrieve the generated text from the model's response.
    var text = new JSONPointer("/results/0/
outputText").queryFrom(responseBody).toString();
    System.out.println(text);

    return text;

} catch (SdkClientException e) {
    System.err.printf("ERROR: Can't invoke '%s'. Reason: %s", modelId,
e.getMessage());
    throw new RuntimeException(e);
}
}
```

```
public static void main(String[] args) {
    invokeModel();
}
}
```

- Per i dettagli sull'API, consulta [InvokeModel](#) in AWS SDK for Java 2.x API Reference.

## InvokeModelWithResponseStream

Il seguente esempio di codice mostra come inviare un messaggio di testo ai modelli Amazon Titan Text, utilizzando l'API Invoke Model, e stampare il flusso di risposta.

### SDK per Java 2.x

#### Note

C'è altro su GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Utilizza l'API Invoke Model per inviare un messaggio di testo ed elaborare il flusso di risposta in tempo reale.

```
// Use the native inference API to send a text message to Amazon Titan Text
// and print the response stream.

import org.json.JSONObject;
import org.json.JSONPointer;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
import
    software.amazon.awssdk.services.bedrockruntime.model.InvokeModelWithResponseStreamRequest;
import
    software.amazon.awssdk.services.bedrockruntime.model.InvokeModelWithResponseStreamResponse;

import java.util.concurrent.ExecutionException;
```

```
import static
software.amazon.awssdk.services.bedrockruntime.model.InvokeModelWithResponseStreamResponseH

public class InvokeModelWithResponseStream {

    public static String invokeModelWithResponseStream() throws ExecutionException,
    InterruptedException {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
    provider.
        var client = BedrockRuntimeAsyncClient.builder()
            .credentialsProvider(DefaultCredentialsProvider.create())
            .region(Region.US_EAST_1)
            .build();

        // Set the model ID, e.g., Titan Text Premier.
        var modelId = "amazon.titan-text-premier-v1:0";

        // The InvokeModelWithResponseStream API uses the model's native payload.
        // Learn more about the available inference parameters and response fields
    at:
        // https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-
    titan-text.html
        var nativeRequestTemplate = "{ \"inputText\": \"{{prompt}}\" }";

        // Define the prompt for the model.
        var prompt = "Describe the purpose of a 'hello world' program in one line.";

        // Embed the prompt in the model's native request payload.
        String nativeRequest = nativeRequestTemplate.replace("{{prompt}}", prompt);

        // Create a request with the model ID and the model's native request
    payload.
        var request = InvokeModelWithResponseStreamRequest.builder()
            .body(SdkBytes.fromUtf8String(nativeRequest))
            .modelId(modelId)
            .build();

        // Prepare a buffer to accumulate the generated response text.
        var completeResponseTextBuffer = new StringBuilder();

        // Prepare a handler to extract, accumulate, and print the response text in
    real-time.
```

```
var responseStreamHandler =
InvokeModelWithResponseStreamResponseHandler.builder()
    .subscriber(Visitor.builder().onChunk(chunk -> {
        // Extract and print the text from the model's native response.
        var response = new JSONObject(chunk.bytes().asUtf8String());
        var text = new JSONPointer("/outputText").queryFrom(response);
        System.out.print(text);

        // Append the text to the response text buffer.
        completeResponseTextBuffer.append(text);
    }).build()).build();

try {
    // Send the request and wait for the handler to process the response.
    client.invokeModelWithResponseStream(request,
responseStreamHandler).get();

    // Return the complete response text.
    return completeResponseTextBuffer.toString();

} catch (ExecutionException | InterruptedException e) {
    System.err.printf("Can't invoke '%s': %s", modelId,
e.getCause().getMessage());
    throw new RuntimeException(e);
}

}

public static void main(String[] args) throws ExecutionException,
InterruptedException {
    invokeModelWithResponseStream();
}
}
```

- Per i dettagli sull'API, consulta la sezione [InvokeModelWithResponseStream AWS SDK for Java 2.x API Reference](#).

## Amazon Titan Text Embeddings

### InvokeModel

L'esempio di codice seguente mostra come:

- Inizia a creare il tuo primo incorporamento.
- Crea incorporamenti configurando il numero di dimensioni e la normalizzazione (solo V2).

## SDK per Java 2.x

### Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

## Crea il tuo primo incorporamento con Titan Text Embeddings V2.

```
// Generate and print an embedding with Amazon Titan Text Embeddings.

import org.json.JSONObject;
import org.json.JSONPointer;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.core.exception.SdkClientException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeClient;

public class InvokeModel {

    public static String invokeModel() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
        provider.
        var client = BedrockRuntimeClient.builder()
            .credentialsProvider(DefaultCredentialsProvider.create())
            .region(Region.US_EAST_1)
            .build();

        // Set the model ID, e.g., Titan Text Embeddings V2.
        var modelId = "amazon.titan-embed-text-v2:0";

        // The InvokeModel API uses the model's native payload.
        // Learn more about the available inference parameters and response fields
        at:
```

```
// https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-
titan-embed-text.html
var nativeRequestTemplate = "{ \"inputText\": \"{{inputText}}\" }";

// The text to convert into an embedding.
var inputText = "Please recommend books with a theme similar to the movie
'Inception'.";

// Embed the prompt in the model's native request payload.
String nativeRequest = nativeRequestTemplate.replace("{{inputText}}",
inputText);

try {
    // Encode and send the request to the Bedrock Runtime.
    var response = client.invokeModel(request -> request
        .body(SdkBytes.fromUtf8String(nativeRequest))
        .modelId(modelId)
    );

    // Decode the response body.
    var responseBody = new JSONObject(response.body().asUtf8String());

    // Retrieve the generated text from the model's response.
    var text = new JSONPointer("/
embedding").queryFrom(responseBody).toString();
    System.out.println(text);

    return text;

} catch (SdkClientException e) {
    System.err.printf("ERROR: Can't invoke '%s'. Reason: %s", modelId,
e.getMessage());
    throw new RuntimeException(e);
}

public static void main(String[] args) {
    invokeModel();
}
}
```

Richiama Titan Text Embeddings V2 per configurare il numero di dimensioni e la normalizzazione.

```
/**
 * Invoke Amazon Titan Text Embeddings V2 with additional inference parameters.
 *
 * @param inputText - The text to convert to an embedding.
 * @param dimensions - The number of dimensions the output embeddings should
have.
 *
 *           Values accepted by the model: 256, 512, 1024.
 * @param normalize - A flag indicating whether or not to normalize the output
embeddings.
 * @return The {@link JSONObject} representing the model's response.
 */
public static JSONObject invokeModel(String inputText, int dimensions, boolean
normalize) {

    // Create a Bedrock Runtime client in the AWS Region of your choice.
    var client = BedrockRuntimeClient.builder()
        .region(Region.US_WEST_2)
        .build();

    // Set the model ID, e.g., Titan Embed Text v2.0.
    var modelId = "amazon.titan-embed-text-v2:0";

    // Create the request for the model.
    var nativeRequest = ""
        {
            "inputText": "%s",
            "dimensions": %d,
            "normalize": %b
        }
        ""formatted(inputText, dimensions, normalize);

    // Encode and send the request.
    var response = client.invokeModel(request -> {
        request.body(SdkBytes.fromUtf8String(nativeRequest));
        request.modelId(modelId);
    });

    // Decode the model's response.
    var modelResponse = new JSONObject(response.body().asUtf8String());

    // Extract and print the generated embedding and the input text token count.
    var embedding = modelResponse.getJSONArray("embedding");
```



```

    var inputTokenCount = modelResponse.getBigInteger("inputTextTokenCount");
    System.out.println("Embedding: " + embedding);
    System.out.println("\nInput token count: " + inputTokenCount);

    // Return the model's native response.
    return modelResponse;
}

```

- Per i dettagli sull'API, consulta la sezione API Reference. [InvokeModel](#) AWS SDK for Java 2.x

## Anthropic Claude

### conversare

Il seguente esempio di codice mostra come inviare un messaggio di testo a Anthropic Claude, utilizzando l'API Converse di Bedrock.

### SDK per Java 2.x

#### Note

C'è di più su. [GitHub](#) Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Invia un messaggio di testo a Anthropic Claude, utilizzando l'API Converse di Bedrock.

```

// Use the Converse API to send a text message to Anthropic Claude.

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.exception.SdkClientException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeClient;
import software.amazon.awssdk.services.bedrockruntime.model.ContentBlock;
import software.amazon.awssdk.services.bedrockruntime.model.ConversationRole;
import software.amazon.awssdk.services.bedrockruntime.model.ConverseResponse;
import software.amazon.awssdk.services.bedrockruntime.model.Message;

public class Converse {

    public static String converse() {

```

```
// Create a Bedrock Runtime client in the AWS Region you want to use.
// Replace the DefaultCredentialsProvider with your preferred credentials
provider.
var client = BedrockRuntimeClient.builder()
    .credentialsProvider(DefaultCredentialsProvider.create())
    .region(Region.US_EAST_1)
    .build();

// Set the model ID, e.g., Claude 3 Haiku.
var modelId = "anthropic.claude-3-haiku-20240307-v1:0";

// Create the input text and embed it in a message object with the user
role.
var inputText = "Describe the purpose of a 'hello world' program in one
line.";
var message = Message.builder()
    .content(ContentBlock.fromText(inputText))
    .role(ConversationRole.USER)
    .build();

try {
    // Send the message with a basic inference configuration.
    ConverseResponse response = client.converse(request -> request
        .modelId(modelId)
        .messages(message)
        .inferenceConfig(config -> config
            .maxTokens(512)
            .temperature(0.5F)
            .topP(0.9F)));

    // Retrieve the generated text from Bedrock's response object.
    var responseText = response.output().message().content().get(0).text();
    System.out.println(responseText);

    return responseText;

} catch (SdkClientException e) {
    System.err.printf("ERROR: Can't invoke '%s'. Reason: %s", modelId,
e.getMessage());
    throw new RuntimeException(e);
}
}
```

```
    public static void main(String[] args) {
        converse();
    }
}
```

Invia un messaggio di testo a Anthropic Claude, utilizzando l'API Converse di Bedrock con il client Java asincrono.

```
// Use the Converse API to send a text message to Anthropic Claude
// with the async Java client.

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
import software.amazon.awssdk.services.bedrockruntime.model.ContentBlock;
import software.amazon.awssdk.services.bedrockruntime.model.ConversationRole;
import software.amazon.awssdk.services.bedrockruntime.model.Message;

import java.util.concurrent.CompletableFuture;
import java.util.concurrent.ExecutionException;

public class ConverseAsync {

    public static String converseAsync() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
        provider.
        var client = BedrockRuntimeAsyncClient.builder()
            .credentialsProvider(DefaultCredentialsProvider.create())
            .region(Region.US_EAST_1)
            .build();

        // Set the model ID, e.g., Claude 3 Haiku.
        var modelId = "anthropic.claude-3-haiku-20240307-v1:0";

        // Create the input text and embed it in a message object with the user
        role.
        var inputText = "Describe the purpose of a 'hello world' program in one
        line.";
        var message = Message.builder()
```

```
        .content(ContentBlock.fromText(inputText))
        .role(ConversationRole.USER)
        .build();

// Send the message with a basic inference configuration.
var request = client.converse(params -> params
    .modelId(modelId)
    .messages(message)
    .inferenceConfig(config -> config
        .maxTokens(512)
        .temperature(0.5F)
        .topP(0.9F))
);

// Prepare a future object to handle the asynchronous response.
CompletableFuture<String> future = new CompletableFuture<>();

// Handle the response or error using the future object.
request.whenComplete((response, error) -> {
    if (error == null) {
        // Extract the generated text from Bedrock's response object.
        String responseText =
response.output().message().content().get(0).text();
        future.complete(responseText);
    } else {
        future.completeExceptionally(error);
    }
});

try {
    // Wait for the future object to complete and retrieve the generated
text.
    String responseText = future.get();
    System.out.println(responseText);

    return responseText;
} catch (ExecutionException | InterruptedException e) {
    System.err.printf("Can't invoke '%s': %s", modelId, e.getMessage());
    throw new RuntimeException(e);
}
}

public static void main(String[] args) {
```

```
        converseAsync();
    }
}
```

- [Per i dettagli sulle API, consulta Converse in API Reference.AWS SDK for Java 2.x](#)

## ConverseStream

Il seguente esempio di codice mostra come inviare un messaggio di testo ad Anthropic Claude, utilizzando l'API Converse di Bedrock ed elaborare il flusso di risposta in tempo reale.

## SDK per Java 2.x

### Note

C'è altro su [GitHub](#) Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Invia un messaggio di testo a Anthropic Claude utilizzando l'API Converse di Bedrock ed elabora il flusso di risposta in tempo reale.

```
// Use the Converse API to send a text message to Anthropic Claude
// and print the response stream.

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
import software.amazon.awssdk.services.bedrockruntime.model.ContentBlock;
import software.amazon.awssdk.services.bedrockruntime.model.ConversationRole;
import
    software.amazon.awssdk.services.bedrockruntime.model.ConverseStreamResponseHandler;
import software.amazon.awssdk.services.bedrockruntime.model.Message;

import java.util.concurrent.ExecutionException;

public class ConverseStream {

    public static void main(String[] args) {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
```

```
// Replace the DefaultCredentialsProvider with your preferred credentials
provider.
var client = BedrockRuntimeAsyncClient.builder()
    .credentialsProvider(DefaultCredentialsProvider.create())
    .region(Region.US_EAST_1)
    .build();

// Set the model ID, e.g., Claude 3 Haiku.
var modelId = "anthropic.claude-3-haiku-20240307-v1:0";

// Create the input text and embed it in a message object with the user
role.
var inputText = "Describe the purpose of a 'hello world' program in one
line.";
var message = Message.builder()
    .content(ContentBlock.fromText(inputText))
    .role(ConversationRole.USER)
    .build();

// Create a handler to extract and print the response text in real-time.
var responseStreamHandler = ConverseStreamResponseHandler.builder()
    .subscriber(ConverseStreamResponseHandler.Visitor.builder()
        .onContentBlockDelta(chunk -> {
            String responseText = chunk.delta().text();
            System.out.print(responseText);
        }).build()
    ).onError(err ->
        System.err.printf("Can't invoke '%s': %s", modelId,
err.getMessage())
    ).build();

try {
    // Send the message with a basic inference configuration and attach the
handler.
    client.converseStream(request -> request.modelId(modelId)
        .messages(message)
        .inferenceConfig(config -> config
            .maxTokens(512)
            .temperature(0.5F)
            .topP(0.9F)
        ), responseStreamHandler).get();

} catch (ExecutionException | InterruptedException e) {
```

```
        System.err.printf("Can't invoke '%s': %s", modelId,
            e.getCause().getMessage());
    }
}
```

- Per i dettagli sull'API, consulta [ConverseStream](#) API Reference. AWS SDK for Java 2.x

## InvokeModel

Il seguente esempio di codice mostra come inviare un messaggio di testo a Anthropic Claude, utilizzando l'API Invoke Model.

### SDK per Java 2.x

#### Note

C'è altro su GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Usa l'API Invoke Model per inviare un messaggio di testo.

```
// Use the native inference API to send a text message to Anthropic Claude.

import org.json.JSONObject;
import org.json.JSONPointer;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.core.exception.SdkClientException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeClient;

public class InvokeModel {

    public static String invokeModel() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
        provider.
        var client = BedrockRuntimeClient.builder()
```

```
        .credentialsProvider(DefaultCredentialsProvider.create())
        .region(Region.US_EAST_1)
        .build();

// Set the model ID, e.g., Claude 3 Haiku.
var modelId = "anthropic.claude-3-haiku-20240307-v1:0";

// The InvokeModel API uses the model's native payload.
// Learn more about the available inference parameters and response fields
at:
// https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-
anthropic-claude-messages.html
var nativeRequestTemplate = """
    {
        "anthropic_version": "bedrock-2023-05-31",
        "max_tokens": 512,
        "temperature": 0.5,
        "messages": [{
            "role": "user",
            "content": "{{prompt}}"
        }]
    }""";

// Define the prompt for the model.
var prompt = "Describe the purpose of a 'hello world' program in one line.";

// Embed the prompt in the model's native request payload.
String nativeRequest = nativeRequestTemplate.replace("{{prompt}}", prompt);

try {
    // Encode and send the request to the Bedrock Runtime.
    var response = client.invokeModel(request -> request
        .body(SdkBytes.fromUtf8String(nativeRequest))
        .modelId(modelId)
    );

    // Decode the response body.
    var responseBody = new JSONObject(response.body().asUtf8String());

    // Retrieve the generated text from the model's response.
    var text = new JSONPointer("/content/0/
text").queryFrom(responseBody).toString();
    System.out.println(text);
}
```



```
        return text;

    } catch (SdkClientException e) {
        System.err.printf("ERROR: Can't invoke '%s'. Reason: %s", modelId,
e.getMessage());
        throw new RuntimeException(e);
    }
}

public static void main(String[] args) {
    invokeModel();
}
}
```

- Per i dettagli sull'API, consulta [InvokeModel](#) in AWS SDK for Java 2.x API Reference.

## InvokeModelWithResponseStream

Il seguente esempio di codice mostra come inviare un messaggio di testo ai modelli Anthropic Claude, utilizzando l'API Invoke Model, e stampare il flusso di risposta.

## SDK per Java 2.x

### Note

C'è altro su [GitHub](#) Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Utilizza l'API Invoke Model per inviare un messaggio di testo ed elaborare il flusso di risposta in tempo reale.

```
// Use the native inference API to send a text message to Anthropic Claude
// and print the response stream.

import org.json.JSONObject;
import org.json.JSONPointer;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
```

```
import
software.amazon.awssdk.services.bedrockruntime.model.InvokeModelWithResponseStreamRequest;
import
software.amazon.awssdk.services.bedrockruntime.model.InvokeModelWithResponseStreamResponse;

import java.util.Objects;
import java.util.concurrent.ExecutionException;

import static
software.amazon.awssdk.services.bedrockruntime.model.InvokeModelWithResponseStreamResponse;

public class InvokeModelWithResponseStream {

    public static String invokeModelWithResponseStream() throws ExecutionException,
InterruptedException {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
provider.
        var client = BedrockRuntimeAsyncClient.builder()
            .credentialsProvider(DefaultCredentialsProvider.create())
            .region(Region.US_EAST_1)
            .build();

        // Set the model ID, e.g., Claude 3 Haiku.
        var modelId = "anthropic.claude-3-haiku-20240307-v1:0";

        // The InvokeModelWithResponseStream API uses the model's native payload.
        // Learn more about the available inference parameters and response fields
at:
        // https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-
anthropic-claude-messages.html
        var nativeRequestTemplate = """
            {
                "anthropic_version": "bedrock-2023-05-31",
                "max_tokens": 512,
                "temperature": 0.5,
                "messages": [{
                    "role": "user",
                    "content": "{{prompt}}"
                }]
            }""";

        // Define the prompt for the model.
```

```
var prompt = "Describe the purpose of a 'hello world' program in one line.";

// Embed the prompt in the model's native request payload.
String nativeRequest = nativeRequestTemplate.replace("{{prompt}}", prompt);

// Create a request with the model ID and the model's native request
payload.
var request = InvokeModelWithResponseStreamRequest.builder()
    .body(SdkBytes.fromUtf8String(nativeRequest))
    .modelId(modelId)
    .build();

// Prepare a buffer to accumulate the generated response text.
var completeResponseTextBuffer = new StringBuilder();

// Prepare a handler to extract, accumulate, and print the response text in
real-time.
var responseStreamHandler =
InvokeModelWithResponseStreamResponseHandler.builder()
    .subscriber(Visitor.builder().onChunk(chunk -> {
        var response = new JSONObject(chunk.bytes().asUtf8String());

        // Extract and print the text from the content blocks.
        if (Objects.equals(response.getString("type"),
"content_block_delta")) {
            var text = new JSONPointer("/delta/
text").queryFrom(response);
            System.out.print(text);

            // Append the text to the response text buffer.
            completeResponseTextBuffer.append(text);
        }
    })).build()).build();

try {
    // Send the request and wait for the handler to process the response.
    client.invokeModelWithResponseStream(request,
responseStreamHandler).get();

    // Return the complete response text.
    return completeResponseTextBuffer.toString();
} catch (ExecutionException | InterruptedException e) {
```

```
        System.err.printf("Can't invoke '%s': %s", modelId,
e.getCause().getMessage());
        throw new RuntimeException(e);
    }
}

public static void main(String[] args) throws ExecutionException,
InterruptedException {
    invokeModelWithResponseStream();
}
}
```

- Per i dettagli sull'API, consulta la sezione [InvokeModelWithResponseStream AWS SDK for Java 2.x API Reference](#).

## Cohere Command

Converse: tutti i modelli

Il seguente esempio di codice mostra come inviare un messaggio di testo a Cohere Command, utilizzando l'API Converse di Bedrock.

SDK per Java 2.x

### Note

C'è di più su [GitHub](#) Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Invia un messaggio di testo a Cohere Command, utilizzando l'API Converse di Bedrock.

```
// Use the Converse API to send a text message to Cohere Command.

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.exception.SdkClientException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeClient;
import software.amazon.awssdk.services.bedrockruntime.model.ContentBlock;
import software.amazon.awssdk.services.bedrockruntime.model.ConversationRole;
import software.amazon.awssdk.services.bedrockruntime.model.ConverseResponse;
```

```
import software.amazon.awssdk.services.bedrockruntime.model.Message;

public class Converse {

    public static String converse() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
        provider.
        var client = BedrockRuntimeClient.builder()
            .credentialsProvider(DefaultCredentialsProvider.create())
            .region(Region.US_EAST_1)
            .build();

        // Set the model ID, e.g., Command R.
        var modelId = "cohere.command-r-v1:0";

        // Create the input text and embed it in a message object with the user
        role.
        var inputText = "Describe the purpose of a 'hello world' program in one
        line.";
        var message = Message.builder()
            .content(ContentBlock.fromText(inputText))
            .role(ConversationRole.USER)
            .build();

        try {
            // Send the message with a basic inference configuration.
            ConverseResponse response = client.converse(request -> request
                .modelId(modelId)
                .messages(message)
                .inferenceConfig(config -> config
                    .maxTokens(512)
                    .temperature(0.5F)
                    .topP(0.9F)));

            // Retrieve the generated text from Bedrock's response object.
            var responseText = response.output().message().content().get(0).text();
            System.out.println(responseText);

            return responseText;

        } catch (SdkClientException e) {
```

```

        System.err.printf("ERROR: Can't invoke '%s'. Reason: %s", modelId,
e.getMessage());
        throw new RuntimeException(e);
    }
}

public static void main(String[] args) {
    converse();
}
}

```

Invia un messaggio di testo a Cohere Command, utilizzando l'API Converse di Bedrock con il client Java asincrono.

```

// Use the Converse API to send a text message to Cohere Command
// with the async Java client.

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
import software.amazon.awssdk.services.bedrockruntime.model.ContentBlock;
import software.amazon.awssdk.services.bedrockruntime.model.ConversationRole;
import software.amazon.awssdk.services.bedrockruntime.model.Message;

import java.util.concurrent.CompletableFuture;
import java.util.concurrent.ExecutionException;

public class ConverseAsync {

    public static String converseAsync() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
provider.
        var client = BedrockRuntimeAsyncClient.builder()
            .credentialsProvider(DefaultCredentialsProvider.create())
            .region(Region.US_EAST_1)
            .build();

        // Set the model ID, e.g., Command R.
        var modelId = "cohere.command-r-v1:0";
    }
}

```

```
// Create the input text and embed it in a message object with the user
role.
var inputText = "Describe the purpose of a 'hello world' program in one
line.";
var message = Message.builder()
    .content(ContentBlock.fromText(inputText))
    .role(ConversationRole.USER)
    .build();

// Send the message with a basic inference configuration.
var request = client.converse(params -> params
    .modelId(modelId)
    .messages(message)
    .inferenceConfig(config -> config
        .maxTokens(512)
        .temperature(0.5F)
        .topP(0.9F))
);

// Prepare a future object to handle the asynchronous response.
CompletableFuture<String> future = new CompletableFuture<>();

// Handle the response or error using the future object.
request.whenComplete((response, error) -> {
    if (error == null) {
        // Extract the generated text from Bedrock's response object.
        String responseText =
response.output().message().content().get(0).text();
        future.complete(responseText);
    } else {
        future.completeExceptionally(error);
    }
});

try {
    // Wait for the future object to complete and retrieve the generated
text.
    String responseText = future.get();
    System.out.println(responseText);

    return responseText;
} catch (ExecutionException | InterruptedException e) {
    System.err.printf("Can't invoke '%s': %s", modelId, e.getMessage());
}
```

```
        throw new RuntimeException(e);
    }
}

public static void main(String[] args) {
    converseAsync();
}
}
```

- [Per i dettagli sulle API, consulta Converse in API Reference.AWS SDK for Java 2.x](#)

## ConverseStream: Tutti i modelli

Il seguente esempio di codice mostra come inviare un messaggio di testo a Cohere Command, utilizzando l'API Converse di Bedrock ed elaborare il flusso di risposta in tempo reale.

### SDK per Java 2.x

#### Note

C'è altro su [GitHub](#) Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Invia un messaggio di testo a Cohere Command, utilizzando l'API Converse di Bedrock ed elabora il flusso di risposta in tempo reale.

```
// Use the Converse API to send a text message to Cohere Command
// and print the response stream.

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
import software.amazon.awssdk.services.bedrockruntime.model.ContentBlock;
import software.amazon.awssdk.services.bedrockruntime.model.ConversationRole;
import
    software.amazon.awssdk.services.bedrockruntime.model.ConverseStreamResponseHandler;
import software.amazon.awssdk.services.bedrockruntime.model.Message;

import java.util.concurrent.ExecutionException;
```



```
public class ConverseStream {

    public static void main(String[] args) {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
        provider.
        var client = BedrockRuntimeAsyncClient.builder()
            .credentialsProvider(DefaultCredentialsProvider.create())
            .region(Region.US_EAST_1)
            .build();

        // Set the model ID, e.g., Command R.
        var modelId = "cohere.command-r-v1:0";

        // Create the input text and embed it in a message object with the user
        role.
        var inputText = "Describe the purpose of a 'hello world' program in one
        line.";
        var message = Message.builder()
            .content(ContentBlock.fromText(inputText))
            .role(ConversationRole.USER)
            .build();

        // Create a handler to extract and print the response text in real-time.
        var responseStreamHandler = ConverseStreamResponseHandler.builder()
            .subscriber(ConverseStreamResponseHandler.Visitor.builder()
                .onContentBlockDelta(chunk -> {
                    String responseText = chunk.delta().text();
                    System.out.print(responseText);
                }).build()
            ).onError(err ->
                System.err.printf("Can't invoke '%s': %s", modelId,
                err.getMessage())
            ).build();

        try {
            // Send the message with a basic inference configuration and attach the
            handler.
            client.converseStream(request -> request.modelId(modelId)
                .messages(message)
                .inferenceConfig(config -> config
                    .maxTokens(512)
                    .temperature(0.5F)
```

```

        .topP(0.9F)
        ), responseStreamHandler).get();

    } catch (ExecutionException | InterruptedException e) {
        System.err.printf("Can't invoke '%s': %s", modelId,
            e.getCause().getMessage());
    }
}
}

```

- Per i dettagli sull'API, consulta API [ConverseStream](#) Reference AWS SDK for Java 2.x .

## InvokeModel: Comando R e R+

Il seguente esempio di codice mostra come inviare un messaggio di testo a Cohere Command R e R+, utilizzando l'API Invoke Model.

### SDK per Java 2.x

#### Note

C'è altro su GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Usa l'API Invoke Model per inviare un messaggio di testo.

```

// Use the native inference API to send a text message to Cohere Command R.

import org.json.JSONObject;
import org.json.JSONPointer;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.core.exception.SdkClientException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeClient;

public class Command_R_InvokeModel {

    public static String invokeModel() {

```

```
// Create a Bedrock Runtime client in the AWS Region you want to use.
// Replace the DefaultCredentialsProvider with your preferred credentials
provider.
var client = BedrockRuntimeClient.builder()
    .credentialsProvider(DefaultCredentialsProvider.create())
    .region(Region.US_EAST_1)
    .build();

// Set the model ID, e.g., Command R.
var modelId = "cohere.command-r-v1:0";

// The InvokeModel API uses the model's native payload.
// Learn more about the available inference parameters and response fields
at:
// https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-
cohere-command-r-plus.html
var nativeRequestTemplate = "{ \"message\": \"{{prompt}}\" }";

// Define the prompt for the model.
var prompt = "Describe the purpose of a 'hello world' program in one line.";

// Embed the prompt in the model's native request payload.
String nativeRequest = nativeRequestTemplate.replace("{{prompt}}", prompt);

try {
    // Encode and send the request to the Bedrock Runtime.
    var response = client.invokeModel(request -> request
        .body(SdkBytes.fromUtf8String(nativeRequest))
        .modelId(modelId)
    );

    // Decode the response body.
    var responseBody = new JSONObject(response.body().asUtf8String());

    // Retrieve the generated text from the model's response.
    var text = new JSONPointer("/text").queryFrom(responseBody).toString();
    System.out.println(text);

    return text;

} catch (SdkClientException e) {
    System.err.printf("ERROR: Can't invoke '%s'. Reason: %s", modelId,
e.getMessage());
    throw new RuntimeException(e);
}
```

```
    }
  }

  public static void main(String[] args) {
    invokeModel();
  }
}
```

- Per i dettagli sull'API, consulta [InvokeModel](#) in AWS SDK for Java 2.x API Reference.

## InvokeModel: Comando e luce di comando

Il seguente esempio di codice mostra come inviare un messaggio di testo a Cohere Command, utilizzando l'API Invoke Model.

### SDK per Java 2.x

#### Note

C'è di più su [GitHub](#) Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Usa l'API Invoke Model per inviare un messaggio di testo.

```
// Use the native inference API to send a text message to Cohere Command.

import org.json.JSONObject;
import org.json.JSONPointer;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.core.exception.SdkClientException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeClient;

public class Command_InvokeModel {

  public static String invokeModel() {

    // Create a Bedrock Runtime client in the AWS Region you want to use.
```

```
// Replace the DefaultCredentialsProvider with your preferred credentials
provider.
var client = BedrockRuntimeClient.builder()
    .credentialsProvider(DefaultCredentialsProvider.create())
    .region(Region.US_EAST_1)
    .build();

// Set the model ID, e.g., Command Light.
var modelId = "cohere.command-light-text-v14";

// The InvokeModel API uses the model's native payload.
// Learn more about the available inference parameters and response fields
at:
// https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-
cohere-command.html
var nativeRequestTemplate = "{ \"prompt\": \"{{prompt}}\" }";

// Define the prompt for the model.
var prompt = "Describe the purpose of a 'hello world' program in one line.";

// Embed the prompt in the model's native request payload.
String nativeRequest = nativeRequestTemplate.replace("{{prompt}}", prompt);

try {
    // Encode and send the request to the Bedrock Runtime.
    var response = client.invokeModel(request -> request
        .body(SdkBytes.fromUtf8String(nativeRequest))
        .modelId(modelId)
    );

    // Decode the response body.
    var responseBody = new JSONObject(response.body().asUtf8String());

    // Retrieve the generated text from the model's response.
    var text = new JSONPointer("/generations/0/
text").queryFrom(responseBody).toString();
    System.out.println(text);

    return text;

} catch (SdkClientException e) {
    System.err.printf("ERROR: Can't invoke '%s'. Reason: %s", modelId,
e.getMessage());
    throw new RuntimeException(e);
}
```

```
    }  
  }  
  
  public static void main(String[] args) {  
    invokeModel();  
  }  
}
```

- Per i dettagli sull'API, consulta [InvokeModel](#) in AWS SDK for Java 2.x API Reference.

### InvokeModelWithResponseStream: Comando R e R+

Il seguente esempio di codice mostra come inviare un messaggio di testo a Cohere Command, utilizzando l'API Invoke Model con un flusso di risposta.

### SDK per Java 2.x

#### Note

C'è di più su [GitHub](#) Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Utilizza l'API Invoke Model per inviare un messaggio di testo ed elaborare il flusso di risposta in tempo reale.

```
// Use the native inference API to send a text message to Cohere Command R  
// and print the response stream.  
  
import org.json.JSONObject;  
import org.json.JSONPointer;  
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;  
import software.amazon.awssdk.core.SdkBytes;  
import software.amazon.awssdk.regions.Region;  
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;  
import  
    software.amazon.awssdk.services.bedrockruntime.model.InvokeModelWithResponseStreamRequest;  
import  
    software.amazon.awssdk.services.bedrockruntime.model.InvokeModelWithResponseStreamResponse;  
  
import java.util.concurrent.ExecutionException;
```

```
import static
software.amazon.awssdk.services.bedrockruntime.model.InvokeModelWithResponseStreamResponseH

public class Command_R_InvokeModelWithResponseStream {

    public static String invokeModelWithResponseStream() throws ExecutionException,
    InterruptedException {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
provider.
        var client = BedrockRuntimeAsyncClient.builder()
            .credentialsProvider(DefaultCredentialsProvider.create())
            .region(Region.US_EAST_1)
            .build();

        // Set the model ID, e.g., Command R.
        var modelId = "cohere.command-r-v1:0";

        // The InvokeModelWithResponseStream API uses the model's native payload.
        // Learn more about the available inference parameters and response fields
at:
        // https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-
cohere-command-r-plus.html
        var nativeRequestTemplate = "{ \"message\": \"{{prompt}}\\" }";

        // Define the prompt for the model.
        var prompt = "Describe the purpose of a 'hello world' program in one line.";

        // Embed the prompt in the model's native request payload.
        String nativeRequest = nativeRequestTemplate.replace("{{prompt}}", prompt);

        // Create a request with the model ID and the model's native request
payload.
        var request = InvokeModelWithResponseStreamRequest.builder()
            .body(SdkBytes.fromUtf8String(nativeRequest))
            .modelId(modelId)
            .build();

        // Prepare a buffer to accumulate the generated response text.
        var completeResponseTextBuffer = new StringBuilder();
```

```

    // Prepare a handler to extract, accumulate, and print the response text in
    // real-time.
    var responseStreamHandler =
    InvokeModelWithResponseStreamResponseHandler.builder()
        .subscriber(Visitor.builder().onChunk(chunk -> {
            // Extract and print the text from the model's native response.
            var response = new JSONObject(chunk.bytes().asUtf8String());
            var text = new JSONPointer("/text").queryFrom(response);
            System.out.print(text);

            // Append the text to the response text buffer.
            completeResponseTextBuffer.append(text);
        })).build()).build();

    try {
        // Send the request and wait for the handler to process the response.
        client.invokeModelWithResponseStream(request,
        responseStreamHandler).get();

        // Return the complete response text.
        return completeResponseTextBuffer.toString();

    } catch (ExecutionException | InterruptedException e) {
        System.err.printf("Can't invoke '%s': %s", modelId,
        e.getCause().getMessage());
        throw new RuntimeException(e);
    }
}

public static void main(String[] args) throws ExecutionException,
InterruptedException {
    invokeModelWithResponseStream();
}
}

```

- Per i dettagli sull'API, consulta la sezione [InvokeModel AWS SDK for Java 2.x API Reference](#).

## InvokeModelWithResponseStream: Comando e luce di comando

Il seguente esempio di codice mostra come inviare un messaggio di testo a Cohere Command, utilizzando l'API Invoke Model con un flusso di risposta.



## SDK per Java 2.x

 Note

C'è di più su. [GitHub](#) Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Utilizza l'API `Invoke Model` per inviare un messaggio di testo ed elaborare il flusso di risposta in tempo reale.

```
// Use the native inference API to send a text message to Cohere Command
// and print the response stream.

import org.json.JSONObject;
import org.json.JSONPointer;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
import
    software.amazon.awssdk.services.bedrockruntime.model.InvokeModelWithResponseStreamRequest;
import
    software.amazon.awssdk.services.bedrockruntime.model.InvokeModelWithResponseStreamResponse;

import java.util.concurrent.ExecutionException;

import static
    software.amazon.awssdk.services.bedrockruntime.model.InvokeModelWithResponseStreamResponse;

public class Command_InvokeModelWithResponseStream {

    public static String invokeModelWithResponseStream() throws ExecutionException,
        InterruptedException {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
        provider.
        var client = BedrockRuntimeAsyncClient.builder()
            .credentialsProvider(DefaultCredentialsProvider.create())
            .region(Region.US_EAST_1)
            .build();
```

```

// Set the model ID, e.g., Command Light.
var modelId = "cohere.command-light-text-v14";

// The InvokeModelWithResponseStream API uses the model's native payload.
// Learn more about the available inference parameters and response fields
at:
// https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-
cohere-command.html
var nativeRequestTemplate = "{ \"prompt\": \"{{prompt}}\" }";

// Define the prompt for the model.
var prompt = "Describe the purpose of a 'hello world' program in one line.";

// Embed the prompt in the model's native request payload.
String nativeRequest = nativeRequestTemplate.replace("{{prompt}}", prompt);

// Create a request with the model ID and the model's native request
payload.
var request = InvokeModelWithResponseStreamRequest.builder()
    .body(SdkBytes.fromUtf8String(nativeRequest))
    .modelId(modelId)
    .build();

// Prepare a buffer to accumulate the generated response text.
var completeResponseTextBuffer = new StringBuilder();

// Prepare a handler to extract, accumulate, and print the response text in
real-time.
var responseStreamHandler =
InvokeModelWithResponseStreamResponseHandler.builder()
    .subscriber(Visitor.builder().onChunk(chunk -> {
        // Extract and print the text from the model's native response.
        var response = new JSONObject(chunk.bytes().asUtf8String());
        var text = new JSONPointer("/generations/0/
text").queryFrom(response);
        System.out.print(text);

        // Append the text to the response text buffer.
        completeResponseTextBuffer.append(text);
    }).build()).build();

try {
    // Send the request and wait for the handler to process the response.

```

```
        client.invokeModelWithResponseStream(request,
responseStreamHandler).get();

        // Return the complete response text.
        return completeResponseTextBuffer.toString();

    } catch (ExecutionException | InterruptedException e) {
        System.err.printf("Can't invoke '%s': %s", modelId,
e.getCause().getMessage());
        throw new RuntimeException(e);
    }
}

public static void main(String[] args) throws ExecutionException,
InterruptedException {
    invokeModelWithResponseStream();
}
}
```

- Per i dettagli sull'API, consulta la sezione [InvokeModel AWS SDK for Java 2.x API Reference](#).

## Meta Llama

Tutti i modelli: Converse API

Il seguente esempio di codice mostra come inviare un messaggio di testo a Meta Llama, utilizzando l'API Converse di Bedrock.

SDK per Java 2.x

### Note

C'è di più su. [GitHub](#) Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Invia un messaggio di testo a Meta Llama utilizzando l'API Converse di Bedrock.

```
// Use the Converse API to send a text message to Meta Llama.

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
```

```
import software.amazon.awssdk.core.exception.SdkClientException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeClient;
import software.amazon.awssdk.services.bedrockruntime.model.ContentBlock;
import software.amazon.awssdk.services.bedrockruntime.model.ConversationRole;
import software.amazon.awssdk.services.bedrockruntime.model.ConverseResponse;
import software.amazon.awssdk.services.bedrockruntime.model.Message;

public class Converse {

    public static String converse() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
        provider.
        var client = BedrockRuntimeClient.builder()
            .credentialsProvider(DefaultCredentialsProvider.create())
            .region(Region.US_EAST_1)
            .build();

        // Set the model ID, e.g., Llama 3 8b Instruct.
        var modelId = "meta.llama3-8b-instruct-v1:0";

        // Create the input text and embed it in a message object with the user
        role.
        var inputText = "Describe the purpose of a 'hello world' program in one
        line.";
        var message = Message.builder()
            .content(ContentBlock.fromText(inputText))
            .role(ConversationRole.USER)
            .build();

        try {
            // Send the message with a basic inference configuration.
            ConverseResponse response = client.converse(request -> request
                .modelId(modelId)
                .messages(message)
                .inferenceConfig(config -> config
                    .maxTokens(512)
                    .temperature(0.5F)
                    .topP(0.9F)));

            // Retrieve the generated text from Bedrock's response object.
```

```

        var responseText = response.output().message().content().get(0).text();
        System.out.println(responseText);

        return responseText;

    } catch (SdkClientException e) {
        System.err.printf("ERROR: Can't invoke '%s'. Reason: %s", modelId,
e.getMessage());
        throw new RuntimeException(e);
    }
}

public static void main(String[] args) {
    converse();
}
}

```

Invia un messaggio di testo a Meta Llama, utilizzando l'API Converse di Bedrock con il client Java asincrono.

```

// Use the Converse API to send a text message to Meta Llama
// with the async Java client.

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
import software.amazon.awssdk.services.bedrockruntime.model.ContentBlock;
import software.amazon.awssdk.services.bedrockruntime.model.ConversationRole;
import software.amazon.awssdk.services.bedrockruntime.model.Message;

import java.util.concurrent.CompletableFuture;
import java.util.concurrent.ExecutionException;

public class ConverseAsync {

    public static String converseAsync() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
        provider.
        var client = BedrockRuntimeAsyncClient.builder()
            .credentialsProvider(DefaultCredentialsProvider.create())

```

```
        .region(Region.US_EAST_1)
        .build();

// Set the model ID, e.g., Llama 3 8b Instruct.
var modelId = "meta.llama3-8b-instruct-v1:0";

// Create the input text and embed it in a message object with the user
role.
var inputText = "Describe the purpose of a 'hello world' program in one
line.";
var message = Message.builder()
    .content(ContentBlock.fromText(inputText))
    .role(ConversationRole.USER)
    .build();

// Send the message with a basic inference configuration.
var request = client.converse(params -> params
    .modelId(modelId)
    .messages(message)
    .inferenceConfig(config -> config
        .maxTokens(512)
        .temperature(0.5F)
        .topP(0.9F))
);

// Prepare a future object to handle the asynchronous response.
CompletableFuture<String> future = new CompletableFuture<>();

// Handle the response or error using the future object.
request.whenComplete((response, error) -> {
    if (error == null) {
        // Extract the generated text from Bedrock's response object.
        String responseText =
response.output().message().content().get(0).text();
        future.complete(responseText);
    } else {
        future.completeExceptionally(error);
    }
});

try {
    // Wait for the future object to complete and retrieve the generated
text.
    String responseText = future.get();
```

```
        System.out.println(responseText);

        return responseText;

    } catch (ExecutionException | InterruptedException e) {
        System.err.printf("Can't invoke '%s': %s", modelId, e.getMessage());
        throw new RuntimeException(e);
    }
}

public static void main(String[] args) {
    converseAsync();
}
}
```

- [Per i dettagli sulle API, consulta Converse in API Reference.AWS SDK for Java 2.x](#)

## ConverseStream: Tutti i modelli

Il seguente esempio di codice mostra come inviare un messaggio di testo a Meta Llama, utilizzando l'API Converse di Bedrock ed elaborare il flusso di risposta in tempo reale.

## SDK per Java 2.x

### Note

C'è altro su [GitHub](#) Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Invia un messaggio di testo a Meta Llama utilizzando l'API Converse di Bedrock ed elabora il flusso di risposta in tempo reale.

```
// Use the Converse API to send a text message to Meta Llama
// and print the response stream.

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
import software.amazon.awssdk.services.bedrockruntime.model.ContentBlock;
import software.amazon.awssdk.services.bedrockruntime.model.ConversationRole;
```

```
import
software.amazon.awssdk.services.bedrockruntime.model.ConverseStreamResponseHandler;
import software.amazon.awssdk.services.bedrockruntime.model.Message;

import java.util.concurrent.ExecutionException;

public class ConverseStream {

    public static void main(String[] args) {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
        provider.
        var client = BedrockRuntimeAsyncClient.builder()
            .credentialsProvider(DefaultCredentialsProvider.create())
            .region(Region.US_EAST_1)
            .build();

        // Set the model ID, e.g., Llama 3 8b Instruct.
        var modelId = "meta.llama3-8b-instruct-v1:0";

        // Create the input text and embed it in a message object with the user
        role.
        var inputText = "Describe the purpose of a 'hello world' program in one
        line.";
        var message = Message.builder()
            .content(ContentBlock.fromText(inputText))
            .role(ConversationRole.USER)
            .build();

        // Create a handler to extract and print the response text in real-time.
        var responseStreamHandler = ConverseStreamResponseHandler.builder()
            .subscriber(ConverseStreamResponseHandler.Visitor.builder()
                .onContentBlockDelta(chunk -> {
                    String responseText = chunk.delta().text();
                    System.out.print(responseText);
                }).build()
            ).onError(err ->
                System.err.printf("Can't invoke '%s': %s", modelId,
err.getMessage())
            ).build();

        try {
```



```

        // Send the message with a basic inference configuration and attach the
handler.
        client.converseStream(request -> request
            .modelId(modelId)
            .messages(message)
            .inferenceConfig(config -> config
                .maxTokens(512)
                .temperature(0.5F)
                .topP(0.9F)
            ), responseStreamHandler).get();

    } catch (ExecutionException | InterruptedException e) {
        System.err.printf("Can't invoke '%s': %s", modelId,
            e.getCause().getMessage());
    }
}
}
}

```

- Per i dettagli sull'API, consulta API [ConverseStream](#) Reference AWS SDK for Java 2.x .

## InvokeModel: Llama 2

Il seguente esempio di codice mostra come inviare un messaggio di testo a Meta Llama 2, utilizzando l'API Invoke Model.

### SDK per Java 2.x

#### Note

C'è altro su [GitHub](#) Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Usa l'API Invoke Model per inviare un messaggio di testo.

```

// Use the native inference API to send a text message to Meta Llama 2.

import org.json.JSONObject;
import org.json.JSONPointer;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.SdkBytes;

```

```
import software.amazon.awssdk.core.exception.SdkClientException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeClient;

public class Llama2_InvokeModel {

    public static String invokeModel() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
        provider.
        var client = BedrockRuntimeClient.builder()
            .credentialsProvider(DefaultCredentialsProvider.create())
            .region(Region.US_EAST_1)
            .build();

        // Set the model ID, e.g., Llama 2 Chat 13B.
        var modelId = "meta.llama2-13b-chat-v1";

        // The InvokeModel API uses the model's native payload.
        // Learn more about the available inference parameters and response fields
        at:
        // https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-
        meta.html
        var nativeRequestTemplate = "{ \"prompt\": \"{{instruction}}\\" }";

        // Define the prompt for the model.
        var prompt = "Describe the purpose of a 'hello world' program in one line.";

        // Embed the prompt in Llama 2's instruction format.
        var instruction = "<s>[INST] {{prompt}} [/INST]\\n".replace("{{prompt}}",
        prompt);

        // Embed the instruction in the the native request payload.
        var nativeRequest = nativeRequestTemplate.replace("{{instruction}}",
        instruction);

        try {
            // Encode and send the request to the Bedrock Runtime.
            var response = client.invokeModel(request -> request
                .body(SdkBytes.fromUtf8String(nativeRequest))
                .modelId(modelId)
            );
        }
    }
}
```

```
// Decode the response body.
var responseBody = new JSONObject(response.body().asUtf8String());

// Retrieve the generated text from the model's response.
var text = new JSONPointer("/
generation").queryFrom(responseBody).toString();
System.out.println(text);

return text;

} catch (SdkClientException e) {
    System.err.printf("ERROR: Can't invoke '%s'. Reason: %s", modelId,
e.getMessage());
    throw new RuntimeException(e);
}
}

public static void main(String[] args) {
    invokeModel();
}
}
```

- Per i dettagli sull'API, consulta [InvokeModel](#) in AWS SDK for Java 2.x API Reference.

### InvokeModel: Llama 3

Il seguente esempio di codice mostra come inviare un messaggio di testo a Meta Llama 3, utilizzando l'API Invoke Model.

### SDK per Java 2.x

#### Note

C'è altro su [GitHub](#) Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Usa l'API Invoke Model per inviare un messaggio di testo.

```
// Use the native inference API to send a text message to Meta Llama 3.
```

```

import org.json.JSONObject;
import org.json.JSONPointer;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.core.exception.SdkClientException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeClient;

public class Llama3_InvokeModel {

    public static String invokeModel() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
        provider.
        var client = BedrockRuntimeClient.builder()
            .credentialsProvider(DefaultCredentialsProvider.create())
            .region(Region.US_EAST_1)
            .build();

        // Set the model ID, e.g., Llama 3 8b Instruct.
        var modelId = "meta.llama3-8b-instruct-v1:0";

        // The InvokeModel API uses the model's native payload.
        // Learn more about the available inference parameters and response fields
        at:
        // https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-
        meta.html
        var nativeRequestTemplate = "{ \"prompt\": \"{{instruction}}\" }";

        // Define the prompt for the model.
        var prompt = "Describe the purpose of a 'hello world' program in one line.";

        // Embed the prompt in Llama 3's instruction format.
        var instruction = (
            "<|begin_of_text|>\n" +
            "<|start_header_id|>user<|end_header_id|>\n" +
            "{{prompt}} <|eot_id|>\n" +
            "<|start_header_id|>assistant<|end_header_id|>\n"
        ).replace("{{prompt}}", prompt);

        // Embed the instruction in the the native request payload.
        var nativeRequest = nativeRequestTemplate.replace("{{instruction}}",
        instruction);
    }
}

```

```
try {
    // Encode and send the request to the Bedrock Runtime.
    var response = client.invokeModel(request -> request
        .body(SdkBytes.fromUtf8String(nativeRequest))
        .modelId(modelId)
    );

    // Decode the response body.
    var responseBody = new JSONObject(response.body().asUtf8String());

    // Retrieve the generated text from the model's response.
    var text = new JSONPointer("/
generation").queryFrom(responseBody).toString();
    System.out.println(text);

    return text;

} catch (SdkClientException e) {
    System.err.printf("ERROR: Can't invoke '%s'. Reason: %s", modelId,
e.getMessage());
    throw new RuntimeException(e);
}

}

public static void main(String[] args) {
    invokeModel();
}
}
```

- Per i dettagli sull'API, consulta [InvokeModel](#) in AWS SDK for Java 2.x API Reference.

## InvokeModelWithResponseStream: Llama 2

Il seguente esempio di codice mostra come inviare un messaggio di testo a Meta Llama 2, utilizzando l'API Invoke Model, e stampare il flusso di risposta.

## SDK per Java 2.x

 Note

C'è di più su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Utilizza l'API Invoke Model per inviare un messaggio di testo ed elaborare il flusso di risposta in tempo reale.

```
// Use the native inference API to send a text message to Meta Llama 2
// and print the response stream.

import org.json.JSONObject;
import org.json.JSONPointer;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
import
    software.amazon.awssdk.services.bedrockruntime.model.InvokeModelWithResponseStreamRequest;
import
    software.amazon.awssdk.services.bedrockruntime.model.InvokeModelWithResponseStreamResponse;

import java.util.concurrent.ExecutionException;

import static
    software.amazon.awssdk.services.bedrockruntime.model.InvokeModelWithResponseStreamResponse;

public class Llama2_InvokeModelWithResponseStream {

    public static String invokeModelWithResponseStream() throws ExecutionException,
        InterruptedException {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
        provider.
        var client = BedrockRuntimeAsyncClient.builder()
            .credentialsProvider(DefaultCredentialsProvider.create())
            .region(Region.US_EAST_1)
            .build();
```

```
// Set the model ID, e.g., Llama 2 Chat 13B.
var modelId = "meta.llama2-13b-chat-v1";

// The InvokeModelWithResponseStream API uses the model's native payload.
// Learn more about the available inference parameters and response fields
at:
// https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-
meta.html
var nativeRequestTemplate = "{ \"prompt\": \"{{instruction}}\" }";

// Define the prompt for the model.
var prompt = "Describe the purpose of a 'hello world' program in one line.";

// Embed the prompt in Llama 2's instruction format.
var instruction = "<s>[INST] {{prompt}} [/INST]\\n".replace("{{prompt}}",
prompt);

// Embed the instruction in the the native request payload.
var nativeRequest = nativeRequestTemplate.replace("{{instruction}}",
instruction);

// Create a request with the model ID and the model's native request
payload.
var request = InvokeModelWithResponseStreamRequest.builder()
    .body(SdkBytes.fromUtf8String(nativeRequest))
    .modelId(modelId)
    .build();

// Prepare a buffer to accumulate the generated response text.
var completeResponseTextBuffer = new StringBuilder();

// Prepare a handler to extract, accumulate, and print the response text in
real-time.
var responseStreamHandler =
InvokeModelWithResponseStreamResponseHandler.builder()
    .subscriber(Visitor.builder().onChunk(chunk -> {
        // Extract and print the text from the model's native response.
        var response = new JSONObject(chunk.bytes().asUtf8String());
        var text = new JSONPointer("/generation").queryFrom(response);
        System.out.print(text);

        // Append the text to the response text buffer.
        completeResponseTextBuffer.append(text);
    })
    )
    .build();
```

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

    try {
        // Send the request and wait for the handler to process the response.
        client.invokeModelWithResponseStream(request,
responseStreamHandler).get();

        // Return the complete response text.
        return completeResponseTextBuffer.toString();

    } catch (ExecutionException | InterruptedException e) {
        System.err.printf("Can't invoke '%s': %s", modelId,
e.getCause().getMessage());
        throw new RuntimeException(e);
    }
}

public static void main(String[] args) throws ExecutionException,
InterruptedException {
    invokeModelWithResponseStream();
}
}
```

- Per i dettagli sull'API, consulta la sezione [InvokeModelWithResponseStream AWS SDK for Java 2.x API Reference](#).

### InvokeModelWithResponseStream: Llama 3

Il seguente esempio di codice mostra come inviare un messaggio di testo a Meta Llama 3, utilizzando l'API Invoke Model, e stampare il flusso di risposta.

### SDK per Java 2.x

#### Note

C'è di più su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).



Utilizza l'API Invoke Model per inviare un messaggio di testo ed elaborare il flusso di risposta in tempo reale.

```
// Use the native inference API to send a text message to Meta Llama 3
// and print the response stream.

import org.json.JSONObject;
import org.json.JSONPointer;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
import
    software.amazon.awssdk.services.bedrockruntime.model.InvokeModelWithResponseStreamRequest;
import
    software.amazon.awssdk.services.bedrockruntime.model.InvokeModelWithResponseStreamResponse;

import java.util.concurrent.ExecutionException;

import static
    software.amazon.awssdk.services.bedrockruntime.model.InvokeModelWithResponseStreamResponse;

public class Llama3_InvokeModelWithResponseStream {

    public static String invokeModelWithResponseStream() throws ExecutionException,
        InterruptedException {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
        provider.
        var client = BedrockRuntimeAsyncClient.builder()
            .credentialsProvider(DefaultCredentialsProvider.create())
            .region(Region.US_EAST_1)
            .build();

        // Set the model ID, e.g., Llama 3 8b Instruct.
        var modelId = "meta.llama3-8b-instruct-v1:0";

        // The InvokeModelWithResponseStream API uses the model's native payload.
        // Learn more about the available inference parameters and response fields
        at:
        // https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-
        meta.html
```

```
var nativeRequestTemplate = "{ \"prompt\": \"{{instruction}}\" }";

// Define the prompt for the model.
var prompt = "Describe the purpose of a 'hello world' program in one line.";

// Embed the prompt in Llama 3's instruction format.
var instruction = (
    "<|begin_of_text|>\n" +
    "<|start_header_id|>user<|end_header_id|>\n" +
    "{{prompt}} <|eot_id|>\n" +
    "<|start_header_id|>assistant<|end_header_id|>\n"
).replace("{{prompt}}", prompt);

// Embed the instruction in the the native request payload.
var nativeRequest = nativeRequestTemplate.replace("{{instruction}}",
instruction);

// Create a request with the model ID and the model's native request
payload.
var request = InvokeModelWithResponseStreamRequest.builder()
    .body(SdkBytes.fromUtf8String(nativeRequest))
    .modelId(modelId)
    .build();

// Prepare a buffer to accumulate the generated response text.
var completeResponseTextBuffer = new StringBuilder();

// Prepare a handler to extract, accumulate, and print the response text in
real-time.
var responseStreamHandler =
InvokeModelWithResponseStreamResponseHandler.builder()
    .subscriber(Visitor.builder().onChunk(chunk -> {
        // Extract and print the text from the model's native response.
        var response = new JSONObject(chunk.bytes().asUtf8String());
        var text = new JSONPointer("/generation").queryFrom(response);
        System.out.print(text);

        // Append the text to the response text buffer.
        completeResponseTextBuffer.append(text);
    }).build()).build();

try {
    // Send the request and wait for the handler to process the response.
```

```
        client.invokeModelWithResponseStream(request,
responseStreamHandler).get();

        // Return the complete response text.
        return completeResponseTextBuffer.toString();

    } catch (ExecutionException | InterruptedException e) {
        System.err.printf("Can't invoke '%s': %s", modelId,
e.getCause().getMessage());
        throw new RuntimeException(e);
    }
}

public static void main(String[] args) throws ExecutionException,
InterruptedException {
    invokeModelWithResponseStream();
}
}
```

- Per i dettagli sull'API, consulta la sezione [InvokeModelWithResponseStream AWS SDK for Java 2.x API Reference](#).

## IA Mistral

conversare

Il seguente esempio di codice mostra come inviare un messaggio di testo a Mistral, utilizzando l'API Converse di Bedrock.

SDK per Java 2.x

### Note

C'è di più su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Invia un messaggio di testo a Mistral utilizzando l'API Converse di Bedrock.

```
// Use the Converse API to send a text message to Mistral.
```

```
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.exception.SdkClientException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeClient;
import software.amazon.awssdk.services.bedrockruntime.model.ContentBlock;
import software.amazon.awssdk.services.bedrockruntime.model.ConversationRole;
import software.amazon.awssdk.services.bedrockruntime.model.ConverseResponse;
import software.amazon.awssdk.services.bedrockruntime.model.Message;

public class Converse {

    public static String converse() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
        provider.
        var client = BedrockRuntimeClient.builder()
            .credentialsProvider(DefaultCredentialsProvider.create())
            .region(Region.US_EAST_1)
            .build();

        // Set the model ID, e.g., Mistral Large.
        var modelId = "mistral.mistral-large-2402-v1:0";

        // Create the input text and embed it in a message object with the user
        role.
        var inputText = "Describe the purpose of a 'hello world' program in one
        line.";
        var message = Message.builder()
            .content(ContentBlock.fromText(inputText))
            .role(ConversationRole.USER)
            .build();

        try {
            // Send the message with a basic inference configuration.
            ConverseResponse response = client.converse(request -> request
                .modelId(modelId)
                .messages(message)
                .inferenceConfig(config -> config
                    .maxTokens(512)
                    .temperature(0.5F)
                    .topP(0.9F)));
        }
    }
}
```

```

        // Retrieve the generated text from Bedrock's response object.
        var responseText = response.output().message().content().get(0).text();
        System.out.println(responseText);

        return responseText;

    } catch (SdkClientException e) {
        System.err.printf("ERROR: Can't invoke '%s'. Reason: %s", modelId,
e.getMessage());
        throw new RuntimeException(e);
    }

}

public static void main(String[] args) {
    converse();
}
}

```

Invia un messaggio di testo a Mistral, utilizzando l'API Converse di Bedrock con il client Java asincrono.

```

// Use the Converse API to send a text message to Mistral
// with the async Java client.

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
import software.amazon.awssdk.services.bedrockruntime.model.ContentBlock;
import software.amazon.awssdk.services.bedrockruntime.model.ConversationRole;
import software.amazon.awssdk.services.bedrockruntime.model.Message;

import java.util.concurrent.CompletableFuture;
import java.util.concurrent.ExecutionException;

public class ConverseAsync {

    public static String converseAsync() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
        provider.

```

```
var client = BedrockRuntimeAsyncClient.builder()
    .credentialsProvider(DefaultCredentialsProvider.create())
    .region(Region.US_EAST_1)
    .build();

// Set the model ID, e.g., Mistral Large.
var modelId = "mistral.mistral-large-2402-v1:0";

// Create the input text and embed it in a message object with the user
role.
var inputText = "Describe the purpose of a 'hello world' program in one
line.";
var message = Message.builder()
    .content(ContentBlock.fromText(inputText))
    .role(ConversationRole.USER)
    .build();

// Send the message with a basic inference configuration.
var request = client.converse(params -> params
    .modelId(modelId)
    .messages(message)
    .inferenceConfig(config -> config
        .maxTokens(512)
        .temperature(0.5F)
        .topP(0.9F))
);

// Prepare a future object to handle the asynchronous response.
CompletableFuture<String> future = new CompletableFuture<>();

// Handle the response or error using the future object.
request.whenComplete((response, error) -> {
    if (error == null) {
        // Extract the generated text from Bedrock's response object.
        String responseText =
response.output().message().content().get(0).text();
        future.complete(responseText);
    } else {
        future.completeExceptionally(error);
    }
});

try {
```

```
        // Wait for the future object to complete and retrieve the generated
text.
        String responseText = future.get();
        System.out.println(responseText);

        return responseText;

    } catch (ExecutionException | InterruptedException e) {
        System.err.printf("Can't invoke '%s': %s", modelId, e.getMessage());
        throw new RuntimeException(e);
    }
}

public static void main(String[] args) {
    converseAsync();
}
}
```

- [Per i dettagli sull'API, consulta Converse in API Reference.AWS SDK for Java 2.x](#)

## ConverseStream

Il seguente esempio di codice mostra come inviare un messaggio di testo a Mistral, utilizzando l'API Converse di Bedrock ed elaborare il flusso di risposta in tempo reale.

## SDK per Java 2.x

### Note

C'è altro su [GitHub](#) Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Invia un messaggio di testo a Mistral utilizzando l'API Converse di Bedrock ed elabora il flusso di risposta in tempo reale.

```
// Use the Converse API to send a text message to Mistral
// and print the response stream.

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.regions.Region;
```

```
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
import software.amazon.awssdk.services.bedrockruntime.model.ContentBlock;
import software.amazon.awssdk.services.bedrockruntime.model.ConversationRole;
import
    software.amazon.awssdk.services.bedrockruntime.model.ConverseStreamResponseHandler;
import software.amazon.awssdk.services.bedrockruntime.model.Message;

import java.util.concurrent.ExecutionException;

public class ConverseStream {

    public static void main(String[] args) {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
        provider.
        var client = BedrockRuntimeAsyncClient.builder()
            .credentialsProvider(DefaultCredentialsProvider.create())
            .region(Region.US_EAST_1)
            .build();

        // Set the model ID, e.g., Mistral Large.
        var modelId = "mistral.mistral-large-2402-v1:0";

        // Create the input text and embed it in a message object with the user
        role.
        var inputText = "Describe the purpose of a 'hello world' program in one
        line.";
        var message = Message.builder()
            .content(ContentBlock.fromText(inputText))
            .role(ConversationRole.USER)
            .build();

        // Create a handler to extract and print the response text in real-time.
        var responseStreamHandler = ConverseStreamResponseHandler.builder()
            .subscriber(ConverseStreamResponseHandler.Visitor.builder()
                .onContentBlockDelta(chunk -> {
                    String responseText = chunk.delta().text();
                    System.out.print(responseText);
                }).build()
            ).onError(err ->
                System.err.printf("Can't invoke '%s': %s", modelId,
                err.getMessage())
            ).build();
```



```
    try {
        // Send the message with a basic inference configuration and attach the
        handler.
        client.converseStream(request -> request.modelId(modelId)
            .messages(message)
            .inferenceConfig(config -> config
                .maxTokens(512)
                .temperature(0.5F)
                .topP(0.9F)
            ), responseStreamHandler).get();

    } catch (ExecutionException | InterruptedException e) {
        System.err.printf("Can't invoke '%s': %s", modelId,
            e.getCause().getMessage());
    }
}
```

- Per i dettagli sull'API, consulta [ConverseStream](#) API Reference. AWS SDK for Java 2.x

## InvokeModel

Il seguente esempio di codice mostra come inviare un messaggio di testo ai modelli Mistral, utilizzando l'API Invoke Model.

## SDK per Java 2.x

### Note

C'è di più su. [GitHub](#) Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Usa l'API Invoke Model per inviare un messaggio di testo.

```
// Use the native inference API to send a text message to Mistral.

import org.json.JSONObject;
import org.json.JSONPointer;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
```

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.core.exception.SdkClientException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeClient;

public class InvokeModel {

    public static String invokeModel() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
        provider.
        var client = BedrockRuntimeClient.builder()
            .credentialsProvider(DefaultCredentialsProvider.create())
            .region(Region.US_EAST_1)
            .build();

        // Set the model ID, e.g., Mistral Large.
        var modelId = "mistral.mistral-large-2402-v1:0";

        // The InvokeModel API uses the model's native payload.
        // Learn more about the available inference parameters and response fields
        at:
        // https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-
        mistral-text-completion.html
        var nativeRequestTemplate = "{ \"prompt\": \"{{instruction}}\" }";

        // Define the prompt for the model.
        var prompt = "Describe the purpose of a 'hello world' program in one line.";

        // Embed the prompt in Mistral's instruction format.
        var instruction = "<s>[INST] {{prompt}} [/INST]\\n".replace("{{prompt}}",
        prompt);

        // Embed the instruction in the the native request payload.
        var nativeRequest = nativeRequestTemplate.replace("{{instruction}}",
        instruction);

        try {
            // Encode and send the request to the Bedrock Runtime.
            var response = client.invokeModel(request -> request
                .body(SdkBytes.fromUtf8String(nativeRequest))
                .modelId(modelId)
            );
        }
    }
}
```

```
        // Decode the response body.
        var responseBody = new JSONObject(response.body().asUtf8String());

        // Retrieve the generated text from the model's response.
        var text = new JSONPointer("/outputs/0/
text").queryFrom(responseBody).toString();
        System.out.println(text);

        return text;

    } catch (SdkClientException e) {
        System.err.printf("ERROR: Can't invoke '%s'. Reason: %s", modelId,
e.getMessage());
        throw new RuntimeException(e);
    }
}

public static void main(String[] args) {
    invokeModel();
}
}
```

- Per i dettagli sull'API, consulta [InvokeModel](#) in AWS SDK for Java 2.x API Reference.

## InvokeModelWithResponseStream

Il seguente esempio di codice mostra come inviare un messaggio di testo ai modelli Mistral AI, utilizzando l'API Invoke Model, e stampare il flusso di risposta.

### SDK per Java 2.x

#### Note

C'è di più su. [GitHub](#) Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Utilizza l'API Invoke Model per inviare un messaggio di testo ed elaborare il flusso di risposta in tempo reale.

```

// Use the native inference API to send a text message to Mistral
// and print the response stream.

import org.json.JSONObject;
import org.json.JSONPointer;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
import
    software.amazon.awssdk.services.bedrockruntime.model.InvokeModelWithResponseStreamRequest;
import
    software.amazon.awssdk.services.bedrockruntime.model.InvokeModelWithResponseStreamResponse;

import java.util.concurrent.ExecutionException;

import static
    software.amazon.awssdk.services.bedrockruntime.model.InvokeModelWithResponseStreamResponse;

public class InvokeModelWithResponseStream {

    public static String invokeModelWithResponseStream() throws ExecutionException,
        InterruptedException {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
        provider.
        var client = BedrockRuntimeAsyncClient.builder()
            .credentialsProvider(DefaultCredentialsProvider.create())
            .region(Region.US_EAST_1)
            .build();

        // Set the model ID, e.g., Mistral Large.
        var modelId = "mistral.mistral-large-2402-v1:0";

        // The InvokeModelWithResponseStream API uses the model's native payload.
        // Learn more about the available inference parameters and response fields
        at:
        // https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-
        mistral-text-completion.html
        var nativeRequestTemplate = "{ \"prompt\": \"{{instruction}}\" }";

        // Define the prompt for the model.

```

```
var prompt = "Describe the purpose of a 'hello world' program in one line.";

// Embed the prompt in Mistral's instruction format.
var instruction = "<s>[INST] {{prompt}} [/INST]\\n".replace("{{prompt}}",
prompt);

// Embed the instruction in the the native request payload.
var nativeRequest = nativeRequestTemplate.replace("{{instruction}}",
instruction);

// Create a request with the model ID and the model's native request
payload.
var request = InvokeModelWithResponseStreamRequest.builder()
    .body(SdkBytes.fromUtf8String(nativeRequest))
    .modelId(modelId)
    .build();

// Prepare a buffer to accumulate the generated response text.
var completeResponseTextBuffer = new StringBuilder();

// Prepare a handler to extract, accumulate, and print the response text in
real-time.
var responseStreamHandler =
InvokeModelWithResponseStreamResponseHandler.builder()
    .subscriber(Visitor.builder().onChunk(chunk -> {
        // Extract and print the text from the model's native response.
        var response = new JSONObject(chunk.bytes().asUtf8String());
        var text = new JSONPointer("/outputs/0/
text").queryFrom(response);
        System.out.print(text);

        // Append the text to the response text buffer.
        completeResponseTextBuffer.append(text);
    }).build()).build());

try {
    // Send the request and wait for the handler to process the response.
    client.invokeModelWithResponseStream(request,
responseStreamHandler).get();

    // Return the complete response text.
    return completeResponseTextBuffer.toString();

} catch (ExecutionException | InterruptedException e) {
```

```
        System.err.printf("Can't invoke '%s': %s", modelId,
e.getCause().getMessage());
        throw new RuntimeException(e);
    }
}

public static void main(String[] args) throws ExecutionException,
InterruptedException {
    invokeModelWithResponseStream();
}
}
```

- Per i dettagli sull'API, consulta la sezione [InvokeModelWithResponseStream AWS SDK for Java 2.x API Reference](#).

## Scenari

Crea un'applicazione playground per interagire con i modelli Amazon Bedrock Foundation

Il seguente esempio di codice mostra come creare parchi giochi per interagire con i modelli di base di Amazon Bedrock attraverso diverse modalità.

### SDK per Java 2.x

Java Foundation Model (FM) Playground è un'applicazione di esempio Spring Boot che mostra come usare Amazon Bedrock con Java. Questo esempio mostra come gli sviluppatori Java possono utilizzare Amazon Bedrock per creare applicazioni generative abilitate all'intelligenza artificiale. Puoi testare e interagire con i modelli Amazon Bedrock Foundation utilizzando i seguenti tre campi da gioco:

- Un parco giochi testuale.
- Un parco giochi per le chat.
- Un parco giochi di immagini.

L'esempio elenca e visualizza anche i modelli di base a cui avete accesso, insieme alle loro caratteristiche. Per il codice sorgente e le istruzioni di distribuzione, consultate il progetto in [GitHub](#).

Servizi utilizzati in questo esempio

- Runtime di Amazon Bedrock

## Diffusione stabile

### InvokeModel

Il seguente esempio di codice mostra come richiamare Stability.ai Stable Diffusion XL su Amazon Bedrock per generare un'immagine.

### SDK per Java 2.x

#### Note

C'è altro su GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Crea un'immagine con Stable Diffusion.

```
// Create an image with Stable Diffusion.

import org.json.JSONObject;
import org.json.JSONPointer;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.core.exception.SdkClientException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeClient;

import java.math.BigInteger;
import java.security.SecureRandom;

import static com.example.bedrockruntime.libs.ImageTools.displayImage;

public class InvokeModel {

    public static String invokeModel() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
        provider.
        var client = BedrockRuntimeClient.builder()
            .credentialsProvider(DefaultCredentialsProvider.create())
            .region(Region.US_EAST_1)
            .build();
```

```
// Set the model ID, e.g., Stable Diffusion XL v1.
var modelId = "stability.stable-diffusion-xl-v1";

// The InvokeModel API uses the model's native payload.
// Learn more about the available inference parameters and response fields
at:
// https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-
diffusion-1-0-text-image.html
var nativeRequestTemplate = """
    {
        "text_prompts": [{ "text": "{{prompt}}" }],
        "style_preset": "{{style}}",
        "seed": {{seed}}
    }""";

// Define the prompt for the image generation.
var prompt = "A stylized picture of a cute old steampunk robot";

// Get a random 32-bit seed for the image generation (max. 4,294,967,295).
var seed = new BigInteger(31, new SecureRandom());

// Choose a style preset.
var style = "cinematic";

// Embed the prompt, seed, and style in the model's native request payload.
String nativeRequest = nativeRequestTemplate
    .replace("{{prompt}}", prompt)
    .replace("{{seed}}", seed.toString())
    .replace("{{style}}", style);

try {
    // Encode and send the request to the Bedrock Runtime.
    var response = client.invokeModel(request -> request
        .body(SdkBytes.fromUtf8String(nativeRequest))
        .modelId(modelId)
    );

    // Decode the response body.
    var responseBody = new JSONObject(response.body().asUtf8String());

    // Retrieve the generated image data from the model's response.
    var base64ImageData = new JSONPointer("/artifacts/0/base64")
        .queryFrom(responseBody)
```



```
        .toString();

        return base64ImageData;

    } catch (SdkClientException e) {
        System.err.printf("ERROR: Can't invoke '%s'. Reason: %s", modelId,
            e.getMessage());
        throw new RuntimeException(e);
    }
}

public static void main(String[] args) {
    System.out.println("Generating image. This may take a few seconds...");

    String base64ImageData = invokeModel();

    displayImage(base64ImageData);
}
}
```

- Per i dettagli sull'API, consulta la sezione [InvokeModel AWS SDK for Java 2.x API Reference](#).

## CloudFront esempi che utilizzano SDK for Java 2.x

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando AWS SDK for Java 2.x with CloudFront.

Le operazioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le operazioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati e negli esempi tra servizi.

Scenari: esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio.

Ogni esempio include un collegamento a GitHub, dove è possibile trovare istruzioni su come configurare ed eseguire il codice nel contesto.

### Argomenti

- [Azioni](#)

- [Scenari](#)

## Azioni

### CreateDistribution

Il seguente esempio di codice mostra come utilizzare `CreateDistribution`.

SDK per Java 2.x

#### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

L'esempio seguente utilizza un bucket Amazon Simple Storage Service (Amazon S3) come origine del contenuto.

Dopo aver creato la distribuzione, il codice crea un messaggio [CloudFrontWaiter](#) di attesa che la distribuzione venga distribuita prima di restituirla.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.internal.waiters.ResponseOrException;
import software.amazon.awssdk.services.cloudfront.CloudFrontClient;
import software.amazon.awssdk.services.cloudfront.model.CreateDistributionResponse;
import software.amazon.awssdk.services.cloudfront.model.Distribution;
import software.amazon.awssdk.services.cloudfront.model.GetDistributionResponse;
import software.amazon.awssdk.services.cloudfront.model.ItemSelection;
import software.amazon.awssdk.services.cloudfront.model.Method;
import software.amazon.awssdk.services.cloudfront.model.ViewerProtocolPolicy;
import software.amazon.awssdk.services.cloudfront.waiters.CloudFrontWaiter;
import software.amazon.awssdk.services.s3.S3Client;

import java.time.Instant;

public class CreateDistribution {

    private static final Logger logger =
        LoggerFactory.getLogger(CreateDistribution.class);
```

```

    public static Distribution createDistribution(CloudFrontClient
cloudFrontClient, S3Client s3Client,
        final String bucketName, final String keyGroupId, final
String originAccessControlId) {

        final String region = s3Client.headBucket(b ->
b.bucket(bucketName)).sdkHttpResponse().headers()
            .get("x-amz-bucket-region").get(0);
        final String originDomain = bucketName + ".s3." + region +
".amazonaws.com";
        String originId = originDomain; // Use the originDomain value for
the originId.

        // The service API requires some deprecated methods, such as
// DefaultCacheBehavior.Builder#minTTL and #forwardedValue.
        CreateDistributionResponse createDistResponse =
cloudFrontClient.createDistribution(builder -> builder
            .distributionConfig(b1 -> b1
                .origins(b2 -> b2
                    .quantity(1)
                    .items(b3 -> b3

.domainName(originDomain)

.id(originId)

.s3OriginConfig(builder4 -> builder4

            .originAccessIdentity(

                ""))

            .originAccessControlId(

                originAccessControlId)))

                .defaultCacheBehavior(b2 -> b2

.viewerProtocolPolicy(ViewerProtocolPolicy.ALLOW_ALL)

.targetOriginId(originId)

                    .minTTL(200L)
                    .forwardedValues(b5

-> b5

```

```

.cookies(cp -> cp
    .forward(ItemSelection.NONE))

.queryString(true))
-> b3
    .trustedKeyGroups(b3

.quantity(1)

.items(keyGroupId)

.enabled(true))
> b4
    .allowedMethods(b4 -

.quantity(2)

.items(Method.HEAD, Method.GET)

.cachedMethods(b5 -> b5
    .quantity(2)
    .items(Method.HEAD,
        Method.GET))))
    .cacheBehaviors(b -> b
        .quantity(1)
        .items(b2 -> b2

.pathPattern("/index.html")

.viewerProtocolPolicy(
    ViewerProtocolPolicy.ALLOW_ALL)

.targetOriginId(originId)

.trustedKeyGroups(b3 -> b3

    .quantity(1)

```

```

        .items(keyGroupId)

        .enabled(true))

.minTTL(200L)

.forwardedValues(b4 -> b4

        .cookies(cp -> cp

                .forward(ItemSelection.NONE))

        .queryString(true))

.allowedMethods(b5 -> b5.quantity(2)

        .items(Method.HEAD,

                Method.GET)

        .cachedMethods(b6 -> b6

                .quantity(2)

                .items(Method.HEAD,

                        Method.GET))))))
        .enabled(true)
        .comment("Distribution built with
java")

.callerReference(Instant.now().toString()));

        final Distribution distribution = createDistResponse.distribution();
        logger.info("Distribution created. DomainName: [{}] Id: [{}]",
distribution.domainName(),
                distribution.id());
        logger.info("Waiting for distribution to be deployed ...");
        try (CloudFrontWaiter cfWaiter =
CloudFrontWaiter.builder().client(cloudFrontClient).build()) {
            ResponseOrException<GetDistributionResponse>
responseOrException = cfWaiter

```

```

        .waitUntilDistributionDeployed(builder ->
builder.id(distribution.id()))
        .matched();
        responseOrException.response()
        .orElseThrow(() -> new
RuntimeException("Distribution not created"));
        logger.info("Distribution deployed. DomainName: [{}] Id:
[{}]", distribution.domainName(),
distribution.id());
    }
    return distribution;
}
}
}

```

- Per i dettagli sull'API, consulta la sezione [CreateDistribution AWS SDK for Java 2.xAPI Reference](#).

## CreateFunction

Il seguente esempio di codice mostra come utilizzare `CreateFunction`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```

import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudfront.CloudFrontClient;
import software.amazon.awssdk.services.cloudfront.model.CloudFrontException;
import software.amazon.awssdk.services.cloudfront.model.CreateFunctionRequest;
import software.amazon.awssdk.services.cloudfront.model.CreateFunctionResponse;
import software.amazon.awssdk.services.cloudfront.model.FunctionConfig;
import software.amazon.awssdk.services.cloudfront.model.FunctionRuntime;
import java.io.InputStream;

/**

```

```
* Before running this Java V2 code example, set up your development
* environment, including your credentials.
*
* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class CreateFunction {

    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <functionName> <filePath>

            Where:
                functionName - The name of the function to create.\s
                filePath - The path to a file that contains the application
logic for the function.\s
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String functionName = args[0];
        String filePath = args[1];
        CloudFrontClient cloudFrontClient = CloudFrontClient.builder()
            .region(Region.AWS_GLOBAL)
            .build();

        String funArn = createNewFunction(cloudFrontClient, functionName, filePath);
        System.out.println("The function ARN is " + funArn);
        cloudFrontClient.close();
    }

    public static String createNewFunction(CloudFrontClient cloudFrontClient, String
functionName, String filePath) {
        try {
            InputStream fileIs =
CreateFunction.class.getClassLoader().getResourceAsStream(filePath);
            SdkBytes functionCode = SdkBytes.fromInputStream(fileIs);
```

```
FunctionConfig config = FunctionConfig.builder()
    .comment("Created by using the CloudFront Java API")
    .runtime(FunctionRuntime.CLOUDFRONT_JS_1_0)
    .build();

CreateFunctionRequest functionRequest = CreateFunctionRequest.builder()
    .name(functionName)
    .functionCode(functionCode)
    .functionConfig(config)
    .build();

CreateFunctionResponse response =
cloudFrontClient.createFunction(functionRequest);
return response.functionSummary().functionMetadata().functionARN();

} catch (CloudFrontException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
return "";
}
}
```

- Per i dettagli sull'API, consulta la [CreateFunction](#) sezione AWS SDK for Java 2.x API Reference.

## CreateKeyGroup

Il seguente esempio di codice mostra come utilizzare `CreateKeyGroup`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Un gruppo di chiavi richiede almeno una chiave pubblica utilizzata per verificare gli URL o i cookie firmati.

```
import org.slf4j.Logger;
```



```
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.cloudfront.CloudFrontClient;

import java.util.UUID;

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

    public static String createKeyGroup(CloudFrontClient cloudFrontClient, String
publicKeyId) {
        String keyGroupId = cloudFrontClient.createKeyGroup(b -> b.keyGroupConfig(c
-> c
            .items(publicKeyId)
            .name("JavaKeyGroup" + UUID.randomUUID()))
            .keyGroup().id());
        logger.info("KeyGroup created with ID: [{}]", keyGroupId);
        return keyGroupId;
    }
}
```

- Per i dettagli sull'API, consulta la sezione [CreateKeyGroup AWS SDK for Java 2.xAPI Reference](#).

## CreatePublicKey

Il seguente esempio di codice mostra come utilizzare `CreatePublicKey`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Il seguente esempio di codice legge una chiave pubblica e la carica su Amazon. CloudFront

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

```
import software.amazon.awssdk.services.cloudfront.CloudFrontClient;
import software.amazon.awssdk.services.cloudfront.model.CreatePublicKeyResponse;
import software.amazon.awssdk.utils.IoUtils;

import java.io.IOException;
import java.io.InputStream;
import java.util.UUID;

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

    public static String createPublicKey(CloudFrontClient cloudFrontClient, String
        publicKeyFileName) {
        try (InputStream is =
            CreatePublicKey.class.getClassLoader().getResourceAsStream(publicKeyFileName)) {
            String publicKeyString = IoUtils.toUtf8String(is);
            CreatePublicKeyResponse createPublicKeyResponse = cloudFrontClient
                .createPublicKey(b -> b.publicKeyConfig(c -> c
                    .name("JavaCreatedPublicKey" + UUID.randomUUID())
                    .encodedKey(publicKeyString)
                    .callerReference(UUID.randomUUID().toString())));
            String createdPublicKeyId = createPublicKeyResponse.publicKey().id();
            logger.info("Public key created with id: [{}]", createdPublicKeyId);
            return createdPublicKeyId;

        } catch (IOException e) {
            throw new RuntimeException(e);
        }
    }
}
```

- Per i dettagli sull'API, consulta la sezione [AWS SDK for Java 2.x API `CreatePublicKey` Reference](#).

## DeleteDistribution

Il seguente esempio di codice mostra come utilizzare `DeleteDistribution`.

## SDK per Java 2.x

 Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Il seguente esempio di codice aggiorna una distribuzione in modalità disattivata, utilizza un cameriere che attende l'implementazione della modifica, quindi elimina la distribuzione.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.internal.waiters.ResponseOrException;
import software.amazon.awssdk.services.cloudfront.CloudFrontClient;
import software.amazon.awssdk.services.cloudfront.model.DeleteDistributionResponse;
import software.amazon.awssdk.services.cloudfront.model.DistributionConfig;
import software.amazon.awssdk.services.cloudfront.model.GetDistributionResponse;
import software.amazon.awssdk.services.cloudfront.waiters.CloudFrontWaiter;

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

    public static void deleteDistribution(final CloudFrontClient
        cloudFrontClient, final String distributionId) {
        // First, disable the distribution by updating it.
        GetDistributionResponse response =
        cloudFrontClient.getDistribution(b -> b
            .id(distributionId));
        String etag = response.eTag();
        DistributionConfig distConfig =
        response.distribution().distributionConfig();

        cloudFrontClient.updateDistribution(builder -> builder
            .id(distributionId)
            .distributionConfig(builder1 -> builder1
                .cacheBehaviors(distConfig.cacheBehaviors())
                .defaultCacheBehavior(distConfig.defaultCacheBehavior())
                .enabled(false))
        );
    }
}
```

```

        .origins(distConfig.origins())
        .comment(distConfig.comment())

    .callerReference(distConfig.callerReference())

    .defaultCacheBehavior(distConfig.defaultCacheBehavior())
        .priceClass(distConfig.priceClass())
        .aliases(distConfig.aliases())
        .logging(distConfig.logging())

    .defaultRootObject(distConfig.defaultRootObject())

    .customErrorResponses(distConfig.customErrorResponses())

    .httpVersion(distConfig.httpVersion())

    .isIPV6Enabled(distConfig.isIPV6Enabled())

    .restrictions(distConfig.restrictions())

    .viewerCertificate(distConfig.viewerCertificate())
        .webACLId(distConfig.webACLId())

    .originGroups(distConfig.originGroups())
        .ifMatch(etag));

        logger.info("Distribution [{}] is DISABLED, waiting for deployment
before deleting ...",
                    distributionId);
        GetDistributionResponse distributionResponse;
        try (CloudFrontWaiter cfWaiter =
CloudFrontWaiter.builder().client(cloudFrontClient).build()) {
            ResponseOrException<GetDistributionResponse>
responseOrException = cfWaiter
                    .waitUntilDistributionDeployed(builder ->
builder.id(distributionId)).matched();
            distributionResponse = responseOrException.response()
                    .orElseThrow(() -> new
RuntimeException("Could not disable distribution"));
        }

        DeleteDistributionResponse deleteDistributionResponse =
cloudFrontClient
                    .deleteDistribution(builder -> builder

```

```

        .id(distributionId)

    .ifMatch(distributionResponse.eTag());
        if (deleteDistributionResponse.sdkHttpResponse().isSuccessful()) {
            logger.info("Distribution [{}] DELETED", distributionId);
        }
    }
}

```

- Per i dettagli sull'API, consulta [DeleteDistribution](#) API Reference. AWS SDK for Java 2.x

## UpdateDistribution

Il seguente esempio di codice mostra come utilizzare `UpdateDistribution`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudfront.CloudFrontClient;
import software.amazon.awssdk.services.cloudfront.model.GetDistributionRequest;
import software.amazon.awssdk.services.cloudfront.model.GetDistributionResponse;
import software.amazon.awssdk.services.cloudfront.model.Distribution;
import software.amazon.awssdk.services.cloudfront.model.DistributionConfig;
import software.amazon.awssdk.services.cloudfront.model.UpdateDistributionRequest;
import software.amazon.awssdk.services.cloudfront.model.CloudFrontException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ModifyDistribution {

```

```
public static void main(String[] args) {
    final String usage = ""

        Usage:
            <id>\s

        Where:
            id - the id value of the distribution.\s
        """;

    if (args.length != 1) {
        System.out.println(usage);
        System.exit(1);
    }

    String id = args[0];
    CloudFrontClient cloudFrontClient = CloudFrontClient.builder()
        .region(Region.AWS_GLOBAL)
        .build();

    modDistribution(cloudFrontClient, id);
    cloudFrontClient.close();
}

public static void modDistribution(CloudFrontClient cloudFrontClient, String
idVal) {
    try {
        // Get the Distribution to modify.
        GetDistributionRequest disRequest = GetDistributionRequest.builder()
            .id(idVal)
            .build();

        GetDistributionResponse response =
cloudFrontClient.getDistribution(disRequest);
        Distribution disObject = response.distribution();
        DistributionConfig config = disObject.distributionConfig();

        // Create a new DistributionConfig object and add new values to comment
and
        // aliases
        DistributionConfig config1 = DistributionConfig.builder()
            .aliases(config.aliases()) // You can pass in new values here
            .comment("New Comment")
            .cacheBehaviors(config.cacheBehaviors())
```

```

        .priceClass(config.priceClass())
        .defaultCacheBehavior(config.defaultCacheBehavior())
        .enabled(config.enabled())
        .callerReference(config.callerReference())
        .logging(config.logging())
        .originGroups(config.originGroups())
        .origins(config.origins())
        .restrictions(config.restrictions())
        .defaultRootObject(config.defaultRootObject())
        .webACLId(config.webACLId())
        .httpVersion(config.httpVersion())
        .viewerCertificate(config.viewerCertificate())
        .customErrorResponses(config.customErrorResponses())
        .build();

        UpdateDistributionRequest updateDistributionRequest =
UpdateDistributionRequest.builder()
        .distributionConfig(config1)
        .id(disObject.id())
        .ifMatch(response.eTag())
        .build();

        cloudFrontClient.updateDistribution(updateDistributionRequest);

    } catch (CloudFrontException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
}

```

- Per i dettagli sull'API, consulta la [UpdateDistribution](#) sezione AWS SDK for Java 2.x API Reference.

## Scenari

### Elimina le risorse per la firma

Il seguente esempio di codice mostra come eliminare le risorse utilizzate per accedere a contenuti con restrizioni in un bucket Amazon Simple Storage Service (Amazon S3).

## SDK per Java 2.x

 Note

C'è altro su GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.cloudfront.CloudFrontClient;
import software.amazon.awssdk.services.cloudfront.model.DeleteKeyGroupResponse;
import
    software.amazon.awssdk.services.cloudfront.model.DeleteOriginAccessControlResponse;
import software.amazon.awssdk.services.cloudfront.model.DeletePublicKeyResponse;
import software.amazon.awssdk.services.cloudfront.model.GetKeyGroupResponse;
import
    software.amazon.awssdk.services.cloudfront.model.GetOriginAccessControlResponse;
import software.amazon.awssdk.services.cloudfront.model.GetPublicKeyResponse;

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

    public static void deleteOriginAccessControl(final CloudFrontClient
        cloudFrontClient,
        final String originAccessControlId) {
        GetOriginAccessControlResponse getResponse = cloudFrontClient
            .getOriginAccessControl(b -> b.id(originAccessControlId));
        DeleteOriginAccessControlResponse deleteResponse =
            cloudFrontClient.deleteOriginAccessControl(builder -> builder
                .id(originAccessControlId)
                .ifMatch(getResponse.eTag()));
        if (deleteResponse.sdkHttpResponse().isSuccessful()) {
            logger.info("Successfully deleted Origin Access Control [{}]",
                originAccessControlId);
        }
    }

    public static void deleteKeyGroup(final CloudFrontClient cloudFrontClient, final
        String keyGroupId) {
```



```
        GetKeyGroupResponse getResponse = cloudFrontClient.getKeyGroup(b ->
b.id(keyGroupId));
        DeleteKeyGroupResponse deleteResponse =
cloudFrontClient.deleteKeyGroup(builder -> builder
                .id(keyGroupId)
                .ifMatch(getResponse.eTag()));
        if (deleteResponse.sdkHttpResponse().isSuccessful()) {
            logger.info("Successfully deleted Key Group [{}]", keyGroupId);
        }
    }

    public static void deletePublicKey(final CloudFrontClient cloudFrontClient,
final String publicKeyId) {
        GetPublicKeyResponse getResponse = cloudFrontClient.getPublicKey(b ->
b.id(publicKeyId));

        DeletePublicKeyResponse deleteResponse =
cloudFrontClient.deletePublicKey(builder -> builder
                .id(publicKeyId)
                .ifMatch(getResponse.eTag()));

        if (deleteResponse.sdkHttpResponse().isSuccessful()) {
            logger.info("Successfully deleted Public Key [{}]", publicKeyId);
        }
    }
}
```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella Documentazione di riferimento delle API AWS SDK for Java 2.x .
  - [DeleteKeyGroup](#)
  - [DeleteOriginAccessControl](#)
  - [DeletePublicKey](#)

## URL e cookie di firma

Il seguente esempio di codice mostra come creare URL e cookie firmati che consentono l'accesso a risorse con restrizioni.

## SDK per Java 2.x

 Note

C'è altro su [GitHub](#) Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Usa la [CannedSignerRequest](#) classe per firmare URL o cookie con una politica predefinita.

```
import software.amazon.awssdk.services.cloudfront.model.CannedSignerRequest;

import java.net.URL;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.time.Instant;
import java.time.temporal.ChronoUnit;

public class CreateCannedPolicyRequest {

    public static CannedSignerRequest createRequestForCannedPolicy(String
distributionDomainName,
        String fileNameToUpload,
        String privateKeyFullPath, String publicKeyId) throws Exception {
        String protocol = "https";
        String resourcePath = "/" + fileNameToUpload;

        String cloudFrontUrl = new URL(protocol, distributionDomainName,
resourcePath).toString();
        Instant expirationDate = Instant.now().plus(7, ChronoUnit.DAYS);
        Path path = Paths.get(privateKeyFullPath);

        return CannedSignerRequest.builder()
            .resourceUrl(cloudFrontUrl)
            .privateKey(path)
            .keyPairId(publicKeyId)
            .expirationDate(expirationDate)
            .build();
    }
}
```

Usa la [CustomSignerRequest](#) classe per firmare URL o cookie con una politica personalizzata. I `activeDate` e `ipRange` sono metodi opzionali.

```
import software.amazon.awssdk.services.cloudfront.model.CustomSignerRequest;

import java.net.URL;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.time.Instant;
import java.time.temporal.ChronoUnit;

public class CreateCustomPolicyRequest {

    public static CustomSignerRequest createRequestForCustomPolicy(String
distributionDomainName,
        String fileNameToUpload,
        String privateKeyFullPath, String publicKeyId) throws Exception {
        String protocol = "https";
        String resourcePath = "/" + fileNameToUpload;

        String cloudFrontUrl = new URL(protocol, distributionDomainName,
resourcePath).toString();
        Instant expireDate = Instant.now().plus(7, ChronoUnit.DAYS);
        // URL will be accessible tomorrow using the signed URL.
        Instant activeDate = Instant.now().plus(1, ChronoUnit.DAYS);
        Path path = Paths.get(privateKeyFullPath);

        return CustomSignerRequest.builder()
            .resourceUrl(cloudFrontUrl)
            .privateKey(path)
            .keyPairId(publicKeyId)
            .expirationDate(expireDate)
            .activeDate(activeDate) // Optional.
            // .ipRange("192.168.0.1/24") // Optional.
            .build();
    }
}
```

L'esempio seguente dimostra l'uso della [CloudFrontUtilities](#) classe per produrre cookie e URL firmati. [Visualizza](#) questo esempio di codice su GitHub

```
import org.slf4j.Logger;
```

```
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.cloudfront.CloudFrontUtilities;
import software.amazon.awssdk.services.cloudfront.cookie.CookiesForCannedPolicy;
import software.amazon.awssdk.services.cloudfront.cookie.CookiesForCustomPolicy;
import software.amazon.awssdk.services.cloudfront.model.CannedSignerRequest;
import software.amazon.awssdk.services.cloudfront.model.CustomSignerRequest;
import software.amazon.awssdk.services.cloudfront.url.SignedUrl;

public class SigningUtilities {
    private static final Logger logger =
    LoggerFactory.getLogger(SigningUtilities.class);
    private static final CloudFrontUtilities cloudFrontUtilities =
    CloudFrontUtilities.create();

    public static SignedUrl signUrlForCannedPolicy(CannedSignerRequest
    cannedSignerRequest) {
        SignedUrl signedUrl =
    cloudFrontUtilities.getSignedUrlWithCannedPolicy(cannedSignerRequest);
        logger.info("Signed URL: [{}]", signedUrl.url());
        return signedUrl;
    }

    public static SignedUrl signUrlForCustomPolicy(CustomSignerRequest
    customSignerRequest) {
        SignedUrl signedUrl =
    cloudFrontUtilities.getSignedUrlWithCustomPolicy(customSignerRequest);
        logger.info("Signed URL: [{}]", signedUrl.url());
        return signedUrl;
    }

    public static CookiesForCannedPolicy
    getCookiesForCannedPolicy(CannedSignerRequest cannedSignerRequest) {
        CookiesForCannedPolicy cookiesForCannedPolicy = cloudFrontUtilities
            .getCookiesForCannedPolicy(cannedSignerRequest);
        logger.info("Cookie EXPIRES header [{}]",
    cookiesForCannedPolicy.expiresHeaderValue());
        logger.info("Cookie KEYPAIR header [{}]",
    cookiesForCannedPolicy.keyPairIdHeaderValue());
        logger.info("Cookie SIGNATURE header [{}]",
    cookiesForCannedPolicy.signatureHeaderValue());
        return cookiesForCannedPolicy;
    }
}
```

```
public static CookiesForCustomPolicy
getCookiesForCustomPolicy(CustomSignerRequest customSignerRequest) {
    CookiesForCustomPolicy cookiesForCustomPolicy = cloudFrontUtilities
        .getCookiesForCustomPolicy(customSignerRequest);
    logger.info("Cookie POLICY header [{}]",
cookiesForCustomPolicy.policyHeaderValue());
    logger.info("Cookie KEYPAIR header [{}]",
cookiesForCustomPolicy.keyPairIdHeaderValue());
    logger.info("Cookie SIGNATURE header [{}]",
cookiesForCustomPolicy.signatureHeaderValue());
    return cookiesForCustomPolicy;
}
}
```

- Per i dettagli sull'API, consulta la [CloudFrontUtilities](#) sezione AWS SDK for Java 2.x API Reference.

## CloudWatch esempi che utilizzano SDK for Java 2.x

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando AWS SDK for Java 2.x with CloudWatch.

Le operazioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le operazioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati e negli esempi tra servizi.

Scenari: esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio.

Ogni esempio include un collegamento a GitHub, dove è possibile trovare istruzioni su come configurare ed eseguire il codice nel contesto.

Nozioni di base

Salve CloudWatch

I seguenti esempi di codice mostrano come iniziare a utilizzare CloudWatch.

## SDK per Java 2.x

 Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudwatch.CloudWatchClient;
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatch.model.ListMetricsRequest;
import software.amazon.awssdk.services.cloudwatch.paginators.ListMetricsIterable;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class HelloService {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <namespace>\s

                Where:
                namespace - The namespace to filter against (for example, AWS/
                EC2).\s

                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String namespace = args[0];
        Region region = Region.US_EAST_1;
        CloudWatchClient cw = CloudWatchClient.builder()
```

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

    listMets(cw, namespace);
    cw.close();
}

public static void listMets(CloudWatchClient cw, String namespace) {
    try {
        ListMetricsRequest request = ListMetricsRequest.builder()
            .namespace(namespace)
            .build();

        ListMetricsIterable listRes = cw.listMetricsPaginator(request);
        listRes.stream()
            .flatMap(r -> r.metrics().stream())
            .forEach(metrics -> System.out.println(" Retrieved metric is: "
+ metrics.metricName()));

    } catch (CloudWatchException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Per i dettagli sull'API, consulta la [ListMetrics](#) sezione AWS SDK for Java 2.x API Reference.

## Argomenti

- [Azioni](#)
- [Scenari](#)

## Azioni

### DeleteAlarms

Il seguente esempio di codice mostra come utilizzare DeleteAlarms.

## SDK per Java 2.x

 Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudwatch.CloudWatchClient;
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatch.model.DeleteAlarmsRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class DeleteAlarm {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <alarmName>

            Where:
                alarmName - An alarm name to delete (for example, MyAlarm).
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String alarmName = args[0];
        Region region = Region.US_EAST_2;
        CloudWatchClient cw = CloudWatchClient.builder()
            .region(region)
```



```
        .build();

        deleteCWAlarm(cw, alarmName);
        cw.close();
    }

    public static void deleteCWAlarm(CloudWatchClient cw, String alarmName) {
        try {
            DeleteAlarmsRequest request = DeleteAlarmsRequest.builder()
                .alarmNames(alarmName)
                .build();

            cw.deleteAlarms(request);
            System.out.printf("Successfully deleted alarm %s", alarmName);

        } catch (CloudWatchException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Per i dettagli sull'API, consulta la [DeleteAlarms](#) sezione AWS SDK for Java 2.x API Reference.

## DeleteAnomalyDetector

Il seguente esempio di codice mostra come utilizzare `DeleteAnomalyDetector`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public static void deleteAnomalyDetector(CloudWatchClient cw, String fileName) {
    try {
        // Read values from the JSON file.
        JsonParser parser = new JsonFactory().createParser(new File(fileName));
```

```
        com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
        String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
        String customMetricName =
rootNode.findValue("customMetricName").asText();

        SingleMetricAnomalyDetector singleMetricAnomalyDetector =
SingleMetricAnomalyDetector.builder()
            .metricName(customMetricName)
            .namespace(customMetricNamespace)
            .stat("Maximum")
            .build();

        DeleteAnomalyDetectorRequest request =
DeleteAnomalyDetectorRequest.builder()
            .singleMetricAnomalyDetector(singleMetricAnomalyDetector)
            .build();

        cw.deleteAnomalyDetector(request);
        System.out.println("Successfully deleted the Anomaly Detector.");

    } catch (CloudWatchException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

- Per i dettagli sull'API, consulta la [DeleteAnomalyDetector](#) sezione AWS SDK for Java 2.x API Reference.

## DeleteDashboards

Il seguente esempio di codice mostra come utilizzare DeleteDashboards.

## SDK per Java 2.x

**Note**

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public static void deleteDashboard(CloudWatchClient cw, String dashboardName) {
    try {
        DeleteDashboardsRequest dashboardsRequest =
DeleteDashboardsRequest.builder()
        .dashboardNames(dashboardName)
        .build();
        cw.deleteDashboards(dashboardsRequest);
        System.out.println(dashboardName + " was successfully deleted.");

    } catch (CloudWatchException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- Per i dettagli sull'API, consulta la [DeleteDashboards](#) sezione AWS SDK for Java 2.x API Reference.

**DescribeAlarmHistory**

Il seguente esempio di codice mostra come utilizzare `DescribeAlarmHistory`.

## SDK per Java 2.x

**Note**

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public static void getAlarmHistory(CloudWatchClient cw, String fileName, String
date) {
    try {
        // Read values from the JSON file.
        JsonParser parser = new JsonFactory().createParser(new File(fileName));
        com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
        String alarmName = rootNode.findValue("exampleAlarmName").asText();

        Instant start = Instant.parse(date);
        Instant endDate = Instant.now();
        DescribeAlarmHistoryRequest historyRequest =
DescribeAlarmHistoryRequest.builder()
            .startDate(start)
            .endDate(endDate)
            .alarmName(alarmName)
            .historyItemType(HistoryItemType.ACTION)
            .build();

        DescribeAlarmHistoryResponse response =
cw.describeAlarmHistory(historyRequest);
        List<AlarmHistoryItem> historyItems = response.alarmHistoryItems();
        if (historyItems.isEmpty()) {
            System.out.println("No alarm history data found for " + alarmName +
".");
        } else {
            for (AlarmHistoryItem item : historyItems) {
                System.out.println("History summary: " + item.historySummary());
                System.out.println("Time stamp: " + item.timestamp());
            }
        }

    } catch (CloudWatchException | IOException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- Per i dettagli sull'API, consulta la [DescribeAlarmHistory](#) sezione AWS SDK for Java 2.x API Reference.

## DescribeAlarms

Il seguente esempio di codice mostra come utilizzare `DescribeAlarms`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public static void describeAlarms(CloudWatchClient cw) {
    try {
        List<AlarmType> typeList = new ArrayList<>();
        typeList.add(AlarmType.METRIC_ALARM);

        DescribeAlarmsRequest alarmsRequest = DescribeAlarmsRequest.builder()
            .alarmTypes(typeList)
            .maxRecords(10)
            .build();

        DescribeAlarmsResponse response = cw.describeAlarms(alarmsRequest);
        List<MetricAlarm> alarmList = response.metricAlarms();
        for (MetricAlarm alarm : alarmList) {
            System.out.println("Alarm name: " + alarm.alarmName());
            System.out.println("Alarm description: " +
alarm.alarmDescription());
        }
    } catch (CloudWatchException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Per i dettagli sull'API, consulta la [DescribeAlarms](#) sezione AWS SDK for Java 2.x API Reference.

## DescribeAlarmsForMetric

Il seguente esempio di codice mostra come utilizzare `DescribeAlarmsForMetric`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public static void checkForMetricAlarm(CloudWatchClient cw, String fileName) {
    try {
        // Read values from the JSON file.
        JsonParser parser = new JsonFactory().createParser(new File(fileName));
        com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
        String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
        String customMetricName =
rootNode.findValue("customMetricName").asText();
        boolean hasAlarm = false;
        int retries = 10;

        DescribeAlarmsForMetricRequest metricRequest =
DescribeAlarmsForMetricRequest.builder()
            .metricName(customMetricName)
            .namespace(customMetricNamespace)
            .build();

        while (!hasAlarm && retries > 0) {
            DescribeAlarmsForMetricResponse response =
cw.describeAlarmsForMetric(metricRequest);
            hasAlarm = response.hasMetricAlarms();
            retries--;
            Thread.sleep(20000);
            System.out.println(".");
        }
        if (!hasAlarm)
            System.out.println("No Alarm state found for " + customMetricName +
" after 10 retries.");
        else
```

```

        System.out.println("Alarm state found for " + customMetricName +
            ".");
    } catch (CloudWatchException | IOException | InterruptedException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

```

- Per i dettagli sull'API, consulta la [DescribeAlarmsForMetric](#) sezione AWS SDK for Java 2.x API Reference.

## DescribeAnomalyDetectors

Il seguente esempio di codice mostra come utilizzare `DescribeAnomalyDetectors`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```

public static void describeAnomalyDetectors(CloudWatchClient cw, String
fileName) {
    try {
        // Read values from the JSON file.
        JsonParser parser = new JsonFactory().createParser(new File(fileName));
        com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
        String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
        String customMetricName =
rootNode.findValue("customMetricName").asText();
        DescribeAnomalyDetectorsRequest detectorsRequest =
DescribeAnomalyDetectorsRequest.builder()
            .maxResults(10)
            .metricName(customMetricName)
            .namespace(customMetricNamespace)

```

```

        .build();

        DescribeAnomalyDetectorsResponse response =
cw.describeAnomalyDetectors(detectorsRequest);
        List<AnomalyDetector> anomalyDetectorList = response.anomalyDetectors();
        for (AnomalyDetector detector : anomalyDetectorList) {
            System.out.println("Metric name: " +
detector.singleMetricAnomalyDetector().metricName());
            System.out.println("State: " + detector.stateValue());
        }

    } catch (CloudWatchException | IOException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

```

- Per i dettagli sull'API, consulta la [DescribeAnomalyDetectors](#) sezione AWS SDK for Java 2.x API Reference.

## DisableAlarmActions

Il seguente esempio di codice mostra come utilizzare `DisableAlarmActions`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudwatch.CloudWatchClient;
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatch.model.DisableAlarmActionsRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.

```



```
*
* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class DisableAlarmActions {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
            <alarmName>

            Where:
            alarmName - An alarm name to disable (for example, MyAlarm).
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String alarmName = args[0];
        Region region = Region.US_EAST_1;
        CloudWatchClient cw = CloudWatchClient.builder()
            .region(region)
            .build();

        disableActions(cw, alarmName);
        cw.close();
    }

    public static void disableActions(CloudWatchClient cw, String alarmName) {
        try {
            DisableAlarmActionsRequest request =
DisableAlarmActionsRequest.builder()
                .alarmNames(alarmName)
                .build();

            cw.disableAlarmActions(request);
            System.out.printf("Successfully disabled actions on alarm %s",
alarmName);

        } catch (CloudWatchException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }
    }
}
```

```
        System.exit(1);
    }
}
```

- Per i dettagli sull'API, consulta la [DisableAlarmActions](#) sezione AWS SDK for Java 2.x API Reference.

## EnableAlarmActions

Il seguente esempio di codice mostra come utilizzare `EnableAlarmActions`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudwatch.CloudWatchClient;
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatch.model.EnableAlarmActionsRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class EnableAlarmActions {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <alarmName>
```

```

        Where:
        alarmName - An alarm name to enable (for example, MyAlarm).
        """;

    if (args.length != 1) {
        System.out.println(usage);
        System.exit(1);
    }

    String alarm = args[0];
    Region region = Region.US_EAST_1;
    CloudWatchClient cw = CloudWatchClient.builder()
        .region(region)
        .build();

    enableActions(cw, alarm);
    cw.close();
}

public static void enableActions(CloudWatchClient cw, String alarm) {
    try {
        EnableAlarmActionsRequest request = EnableAlarmActionsRequest.builder()
            .alarmNames(alarm)
            .build();

        cw.enableAlarmActions(request);
        System.out.printf("Successfully enabled actions on alarm %s", alarm);

    } catch (CloudWatchException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
}

```

- Per i dettagli sull'API, consulta la [EnableAlarmActions](#) sezione AWS SDK for Java 2.x API Reference.

## GetMetricData

Il seguente esempio di codice mostra come utilizzare `GetMetricData`.

## SDK per Java 2.x

 Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public static void getCustomMetricData(CloudWatchClient cw, String fileName) {
    try {
        // Read values from the JSON file.
        JsonParser parser = new JsonFactory().createParser(new File(fileName));
        com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
        String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
        String customMetricName =
rootNode.findValue("customMetricName").asText();

        // Set the date.
        Instant nowDate = Instant.now();

        long hours = 1;
        long minutes = 30;
        Instant date2 = nowDate.plus(hours, ChronoUnit.HOURS).plus(minutes,
ChronoUnit.MINUTES);

        Metric met = Metric.builder()
            .metricName(customMetricName)
            .namespace(customMetricNamespace)
            .build();

        MetricStat metStat = MetricStat.builder()
            .stat("Maximum")
            .period(1)
            .metric(met)
            .build();

        MetricDataQuery dataQuery = MetricDataQuery.builder()
            .metricStat(metStat)
            .id("foo2")
            .returnData(true)
```

```
        .build();

List<MetricDataQuery> dq = new ArrayList<>();
dq.add(dataQuery);

GetMetricDataRequest getMetReq = GetMetricDataRequest.builder()
    .maxDatapoints(10)
    .scanBy(ScanBy.TIMESTAMP_DESCENDING)
    .startTime(nowDate)
    .endTime(date2)
    .metricDataQueries(dq)
    .build();

GetMetricDataResponse response = cw.getMetricData(getMetReq);
List<MetricDataResult> data = response.metricDataResults();
for (MetricDataResult item : data) {
    System.out.println("The label is " + item.label());
    System.out.println("The status code is " +
item.statusCode().toString());
}

} catch (CloudWatchException | IOException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
}
```

- Per i dettagli sull'API, consulta la [GetMetricData](#) sezione AWS SDK for Java 2.x API Reference.

## GetMetricStatistics

Il seguente esempio di codice mostra come utilizzare `GetMetricStatistics`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```

public static void getAndDisplayMetricStatistics(CloudWatchClient cw, String
nameSpace, String metVal,
        String metricOption, String date, Dimension myDimension) {
    try {
        Instant start = Instant.parse(date);
        Instant endDate = Instant.now();

        GetMetricStatisticsRequest statisticsRequest =
GetMetricStatisticsRequest.builder()
        .endTime(endDate)
        .startTime(start)
        .dimensions(myDimension)
        .metricName(metVal)
        .namespace(nameSpace)
        .period(86400)
        .statistics(Statistic.fromValue(metricOption))
        .build();

        GetMetricStatisticsResponse response =
cw.getMetricStatistics(statisticsRequest);
        List<Datapoint> data = response.datapoints();
        if (!data.isEmpty()) {
            for (Datapoint datapoint : data) {
                System.out
                    .println("Timestamp: " + datapoint.timestamp() + "
Maximum value: " + datapoint.maximum());
            }
        } else {
            System.out.println("The returned data list is empty");
        }

    } catch (CloudWatchException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

```

- Per i dettagli sull'API, consulta la [GetMetricStatistics](#) sezione AWS SDK for Java 2.x API Reference.

## GetMetricWidgetImage

Il seguente esempio di codice mostra come utilizzare `GetMetricWidgetImage`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public static void getAndOpenMetricImage(CloudWatchClient cw, String fileName) {
    System.out.println("Getting Image data for custom metric.");
    try {
        String myJSON = "{\n" +
            "  \"title\": \"Example Metric Graph\",\n" +
            "  \"view\": \"timeSeries\",\n" +
            "  \"stacked\": false,\n" +
            "  \"period\": 10,\n" +
            "  \"width\": 1400,\n" +
            "  \"height\": 600,\n" +
            "  \"metrics\": [\n" +
            "    [\n" +
            "      \"AWS/Billing\",\n" +
            "      \"EstimatedCharges\",\n" +
            "      \"Currency\",\n" +
            "      \"USD\"\n" +
            "    ]\n" +
            "  ]\n" +
            "}";

        GetMetricWidgetImageRequest imageRequest =
            GetMetricWidgetImageRequest.builder()
                .metricWidget(myJSON)
                .build();

        GetMetricWidgetImageResponse response =
            cw.getMetricWidgetImage(imageRequest);
        SdkBytes sdkBytes = response.metricWidgetImage();
        byte[] bytes = sdkBytes.asByteArray();
        File outputFile = new File(fileName);
        try (FileOutputStream outputStream = new FileOutputStream(outputFile)) {
```

```

        outputStream.write(bytes);
    }

    } catch (CloudWatchException | IOException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

```

- Per i dettagli sull'API, consulta la [GetMetricWidgetImage](#) sezione AWS SDK for Java 2.x API Reference.

## ListDashboards

Il seguente esempio di codice mostra come utilizzare `ListDashboards`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```

public static void listDashboards(CloudWatchClient cw) {
    try {
        ListDashboardsIterable listRes = cw.listDashboardsPaginator();
        listRes.stream()
            .flatMap(r -> r.dashboardEntries().stream())
            .forEach(entry -> {
                System.out.println("Dashboard name is: " +
entry.dashboardName());
                System.out.println("Dashboard ARN is: " +
entry.dashboardArn());
            });
    } catch (CloudWatchException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

```



```
}
```

- Per i dettagli sull'API, consulta la [ListDashboards](#) sezione AWS SDK for Java 2.x API Reference.

## ListMetrics

Il seguente esempio di codice mostra come utilizzare `ListMetrics`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudwatch.CloudWatchClient;
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatch.model.ListMetricsRequest;
import software.amazon.awssdk.services.cloudwatch.model.ListMetricsResponse;
import software.amazon.awssdk.services.cloudwatch.model.Metric;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListMetrics {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <namespace>\s

                Where:
```

```
        namespace - The namespace to filter against (for example, AWS/
EC2).\s
        """;

    if (args.length != 1) {
        System.out.println(usage);
        System.exit(1);
    }

    String namespace = args[0];
    Region region = Region.US_EAST_1;
    CloudWatchClient cw = CloudWatchClient.builder()
        .region(region)
        .build();

    listMets(cw, namespace);
    cw.close();
}

public static void listMets(CloudWatchClient cw, String namespace) {
    boolean done = false;
    String nextToken = null;

    try {
        while (!done) {

            ListMetricsResponse response;
            if (nextToken == null) {
                ListMetricsRequest request = ListMetricsRequest.builder()
                    .namespace(namespace)
                    .build();

                response = cw.listMetrics(request);
            } else {
                ListMetricsRequest request = ListMetricsRequest.builder()
                    .namespace(namespace)
                    .nextToken(nextToken)
                    .build();

                response = cw.listMetrics(request);
            }

            for (Metric metric : response.metrics()) {
                System.out.printf("Retrieved metric %s", metric.metricName());
            }
        }
    }
}
```

```

        System.out.println();
    }

    if (response.nextToken() == null) {
        done = true;
    } else {
        nextToken = response.nextToken();
    }
}

} catch (CloudWatchException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
}

```

- Per i dettagli sull'API, consulta la [ListMetrics](#) sezione AWS SDK for Java 2.x API Reference.

## PutAnomalyDetector

Il seguente esempio di codice mostra come utilizzare `PutAnomalyDetector`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```

public static void addAnomalyDetector(CloudWatchClient cw, String fileName) {
    try {
        // Read values from the JSON file.
        JsonParser parser = new JsonFactory().createParser(new File(fileName));
        com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
        String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
        String customMetricName =
rootNode.findValue("customMetricName").asText();

```

```

        SingleMetricAnomalyDetector singleMetricAnomalyDetector =
SingleMetricAnomalyDetector.builder()
        .metricName(customMetricName)
        .namespace(customMetricNamespace)
        .stat("Maximum")
        .build();

        PutAnomalyDetectorRequest anomalyDetectorRequest =
PutAnomalyDetectorRequest.builder()
        .singleMetricAnomalyDetector(singleMetricAnomalyDetector)
        .build();

        cw.putAnomalyDetector(anomalyDetectorRequest);
        System.out.println("Added anomaly detector for metric " +
customMetricName + ".");

    } catch (CloudWatchException | IOException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

```

- Per i dettagli sull'API, consulta la [PutAnomalyDetector](#) sezione AWS SDK for Java 2.x API Reference.

## PutDashboard

Il seguente esempio di codice mostra come utilizzare PutDashboard.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```

public static void createDashboardWithMetrics(CloudWatchClient cw, String
dashboardName, String fileName) {

```

```
try {
    PutDashboardRequest dashboardRequest = PutDashboardRequest.builder()
        .dashboardName(dashboardName)
        .dashboardBody(readFileAsString(fileName))
        .build();

    PutDashboardResponse response = cw.putDashboard(dashboardRequest);
    System.out.println(dashboardName + " was successfully created.");
    List<DashboardValidationMessage> messages =
response.dashboardValidationMessages();
    if (messages.isEmpty()) {
        System.out.println("There are no messages in the new Dashboard");
    } else {
        for (DashboardValidationMessage message : messages) {
            System.out.println("Message is: " + message.message());
        }
    }

} catch (CloudWatchException | IOException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
}
```

- Per i dettagli sull'API, consulta la [PutDashboard](#) sezione AWS SDK for Java 2.x API Reference.

## PutMetricAlarm

Il seguente esempio di codice mostra come utilizzare `PutMetricAlarm`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public static String createAlarm(CloudWatchClient cw, String fileName) {
    try {
        // Read values from the JSON file.
```

```
        JsonParser parser = new JsonFactory().createParser(new File(fileName));
        com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
        String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
        String customMetricName =
rootNode.findValue("customMetricName").asText();
        String alarmName = rootNode.findValue("exampleAlarmName").asText();
        String emailTopic = rootNode.findValue("emailTopic").asText();
        String accountId = rootNode.findValue("accountId").asText();
        String region = rootNode.findValue("region").asText();

        // Create a List for alarm actions.
        List<String> alarmActions = new ArrayList<>();
        alarmActions.add("arn:aws:sns:" + region + ":" + accountId + ":" +
emailTopic);
        PutMetricAlarmRequest alarmRequest = PutMetricAlarmRequest.builder()
            .alarmActions(alarmActions)
            .alarmDescription("Example metric alarm")
            .alarmName(alarmName)

.comparisonOperator(ComparisonOperator.GREATER_THAN_OR_EQUAL_TO_THRESHOLD)
            .threshold(100.00)
            .metricName(customMetricName)
            .namespace(customMetricNamespace)
            .evaluationPeriods(1)
            .period(10)
            .statistic("Maximum")
            .datapointsToAlarm(1)
            .treatMissingData("ignore")
            .build();

        cw.putMetricAlarm(alarmRequest);
        System.out.println(alarmName + " was successfully created!");
        return alarmName;

    } catch (CloudWatchException | IOException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return "";
}
```

- Per i dettagli sull'API, consulta la [PutMetricAlarm](#) sezione AWS SDK for Java 2.x API Reference.

## PutMetricData

Il seguente esempio di codice mostra come utilizzare `PutMetricData`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public static void addMetricDataForAlarm(CloudWatchClient cw, String fileName) {
    try {
        // Read values from the JSON file.
        JsonParser parser = new JsonFactory().createParser(new File(fileName));
        com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
        String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
        String customMetricName =
rootNode.findValue("customMetricName").asText();

        // Set an Instant object.
        String time =
ZonedDateTime.now(ZoneOffset.UTC).format(DateTimeFormatter.ISO_INSTANT);
        Instant instant = Instant.parse(time);

        MetricDatum datum = MetricDatum.builder()
            .metricName(customMetricName)
            .unit(StandardUnit.NONE)
            .value(1001.00)
            .timestamp(instant)
            .build();

        MetricDatum datum2 = MetricDatum.builder()
            .metricName(customMetricName)
            .unit(StandardUnit.NONE)
            .value(1002.00)
            .timestamp(instant)
```

```
        .build();

        List<MetricDatum> metricDataList = new ArrayList<>();
        metricDataList.add(datum);
        metricDataList.add(datum2);

        PutMetricDataRequest request = PutMetricDataRequest.builder()
            .namespace(customMetricNamespace)
            .metricData(metricDataList)
            .build();

        cw.putMetricData(request);
        System.out.println("Added metric values for for metric " +
            customMetricName);

    } catch (CloudWatchException | IOException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- Per i dettagli sull'API, consulta la [PutMetricData](#) sezione AWS SDK for Java 2.x API Reference.

## Scenari

Inizia con parametri, pannelli di controllo e allarmi

L'esempio di codice seguente mostra come:

- Elenca i CloudWatch namespace e le metriche.
- Ottieni le statistiche per un parametro e per la fatturazione stimata.
- Crea e aggiorna un pannello di controllo.
- Crea e aggiungi i dati a un parametro.
- Crea e attiva un allarme, quindi visualizza la cronologia degli allarmi.
- Aggiungi un rilevatore di anomalie.
- Acquisisci uno schema di parametri, quindi elimina le risorse.



## SDK per Java 2.x

 Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import com.fasterxml.jackson.core.JsonFactory;
import com.fasterxml.jackson.core.JsonParser;
import com.fasterxml.jackson.databind.ObjectMapper;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudwatch.CloudWatchClient;
import software.amazon.awssdk.services.cloudwatch.model.AlarmHistoryItem;
import software.amazon.awssdk.services.cloudwatch.model.AlarmType;
import software.amazon.awssdk.services.cloudwatch.model.AnomalyDetector;
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatch.model.ComparisonOperator;
import software.amazon.awssdk.services.cloudwatch.model.DashboardValidationMessage;
import software.amazon.awssdk.services.cloudwatch.model.Datapoint;
import software.amazon.awssdk.services.cloudwatch.model.DeleteAlarmsRequest;
import
    software.amazon.awssdk.services.cloudwatch.model.DeleteAnomalyDetectorRequest;
import software.amazon.awssdk.services.cloudwatch.model.DeleteDashboardsRequest;
import software.amazon.awssdk.services.cloudwatch.model.DescribeAlarmHistoryRequest;
import
    software.amazon.awssdk.services.cloudwatch.model.DescribeAlarmHistoryResponse;
import
    software.amazon.awssdk.services.cloudwatch.model.DescribeAlarmsForMetricRequest;
import
    software.amazon.awssdk.services.cloudwatch.model.DescribeAlarmsForMetricResponse;
import software.amazon.awssdk.services.cloudwatch.model.DescribeAlarmsRequest;
import software.amazon.awssdk.services.cloudwatch.model.DescribeAlarmsResponse;
import
    software.amazon.awssdk.services.cloudwatch.model.DescribeAnomalyDetectorsRequest;
import
    software.amazon.awssdk.services.cloudwatch.model.DescribeAnomalyDetectorsResponse;
import software.amazon.awssdk.services.cloudwatch.model.Dimension;
import software.amazon.awssdk.services.cloudwatch.model.GetMetricDataRequest;
import software.amazon.awssdk.services.cloudwatch.model.GetMetricDataResponse;
```

```
import software.amazon.awssdk.services.cloudwatch.model.GetMetricStatisticsRequest;
import software.amazon.awssdk.services.cloudwatch.model.GetMetricStatisticsResponse;
import software.amazon.awssdk.services.cloudwatch.model.GetMetricWidgetImageRequest;
import
    software.amazon.awssdk.services.cloudwatch.model.GetMetricWidgetImageResponse;
import software.amazon.awssdk.services.cloudwatch.model.HistoryItemType;
import software.amazon.awssdk.services.cloudwatch.model.ListMetricsRequest;
import software.amazon.awssdk.services.cloudwatch.model.ListMetricsResponse;
import software.amazon.awssdk.services.cloudwatch.model.Metric;
import software.amazon.awssdk.services.cloudwatch.model.MetricAlarm;
import software.amazon.awssdk.services.cloudwatch.model.MetricDataQuery;
import software.amazon.awssdk.services.cloudwatch.model.MetricDataResult;
import software.amazon.awssdk.services.cloudwatch.model.MetricDatum;
import software.amazon.awssdk.services.cloudwatch.model.MetricStat;
import software.amazon.awssdk.services.cloudwatch.model.PutAnomalyDetectorRequest;
import software.amazon.awssdk.services.cloudwatch.model.PutDashboardRequest;
import software.amazon.awssdk.services.cloudwatch.model.PutDashboardResponse;
import software.amazon.awssdk.services.cloudwatch.model.PutMetricAlarmRequest;
import software.amazon.awssdk.services.cloudwatch.model.PutMetricDataRequest;
import software.amazon.awssdk.services.cloudwatch.model.ScanBy;
import software.amazon.awssdk.services.cloudwatch.model.SingleMetricAnomalyDetector;
import software.amazon.awssdk.services.cloudwatch.model.StandardUnit;
import software.amazon.awssdk.services.cloudwatch.model.Statistic;
import software.amazon.awssdk.services.cloudwatch.paginators.ListDashboardsIterable;
import software.amazon.awssdk.services.cloudwatch.paginators.ListMetricsIterable;
import java.io.BufferedReader;
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStreamReader;
import java.nio.file.Files;
import java.nio.file.Paths;
import java.time.Instant;
import java.time.ZoneOffset;
import java.time.ZonedDateTime;
import java.time.format.DateTimeFormatter;
import java.time.temporal.ChronoUnit;
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
```

```

*
* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*
* To enable billing metrics and statistics for this example, make sure billing
* alerts are enabled for your account:
* https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/monitor\_estimated\_charges\_with\_cloudwatch.html#turning\_on\_billing\_metrics
*
* This Java code example performs the following tasks:
*
* 1. List available namespaces from Amazon CloudWatch.
* 2. List available metrics within the selected Namespace.
* 3. Get statistics for the selected metric over the last day.
* 4. Get CloudWatch estimated billing for the last week.
* 5. Create a new CloudWatch dashboard with metrics.
* 6. List dashboards using a paginator.
* 7. Create a new custom metric by adding data for it.
* 8. Add the custom metric to the dashboard.
* 9. Create an alarm for the custom metric.
* 10. Describe current alarms.
* 11. Get current data for the new custom metric.
* 12. Push data into the custom metric to trigger the alarm.
* 13. Check the alarm state using the action DescribeAlarmsForMetric.
* 14. Get alarm history for the new alarm.
* 15. Add an anomaly detector for the custom metric.
* 16. Describe current anomaly detectors.
* 17. Get a metric image for the custom metric.
* 18. Clean up the Amazon CloudWatch resources.
*/
public class CloudWatchScenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    public static void main(String[] args) throws IOException {
        final String usage = ""

            Usage:
                <myDate> <costDateWeek> <dashboardName> <dashboardJson>
<dashboardAdd> <settings> <metricImage> \s

            Where:
                myDate - The start date to use to get metric statistics. (For
example, 2023-01-11T18:35:24.00Z.)\s

```

```

        costDateWeek - The start date to use to get AWS/Billinget
statistics. (For example, 2023-01-11T18:35:24.00Z.)\s
        dashboardName - The name of the dashboard to create.\s
        dashboardJson - The location of a JSON file to use to create a
dashboard. (See Readme file.)\s
        dashboardAdd - The location of a JSON file to use to update a
dashboard. (See Readme file.)\s
        settings - The location of a JSON file from which various values
are read. (See Readme file.)\s
        metricImage - The location of a BMP file that is used to create a
graph.\s
        """;

    if (args.length != 7) {
        System.out.println(usage);
        System.exit(1);
    }

    Region region = Region.US_EAST_1;
    String myDate = args[0];
    String costDateWeek = args[1];
    String dashboardName = args[2];
    String dashboardJson = args[3];
    String dashboardAdd = args[4];
    String settings = args[5];
    String metricImage = args[6];

    Double dataPoint = Double.parseDouble("10.0");
    Scanner sc = new Scanner(System.in);
    CloudWatchClient cw = CloudWatchClient.builder()
        .region(region)
        .credentialsProvider(ProfileCredentialsProvider.create())
        .build();

    System.out.println(DASHES);
    System.out.println("Welcome to the Amazon CloudWatch example scenario.");
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println(
        "1. List at least five available unique namespaces from Amazon
CloudWatch. Select one from the list.");
    ArrayList<String> list = listNameSpaces(cw);
    for (int z = 0; z < 5; z++) {

```

```
        int index = z + 1;
        System.out.println("    " + index + ". " + list.get(z));
    }

    String selectedNamespace = "";
    String selectedMetrics = "";
    int num = Integer.parseInt(sc.nextLine());
    if (1 <= num && num <= 5) {
        selectedNamespace = list.get(num - 1);
    } else {
        System.out.println("You did not select a valid option.");
        System.exit(1);
    }
    System.out.println("You selected " + selectedNamespace);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("2. List available metrics within the selected namespace
and select one from the list.");
    ArrayList<String> metList = listMets(cw, selectedNamespace);
    for (int z = 0; z < 5; z++) {
        int index = z + 1;
        System.out.println("    " + index + ". " + metList.get(z));
    }
    num = Integer.parseInt(sc.nextLine());
    if (1 <= num && num <= 5) {
        selectedMetrics = metList.get(num - 1);
    } else {
        System.out.println("You did not select a valid option.");
        System.exit(1);
    }
    System.out.println("You selected " + selectedMetrics);
    Dimension myDimension = getSpecificMet(cw, selectedNamespace);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("3. Get statistics for the selected metric over the last
day.");
    String metricOption = "";
    ArrayList<String> statTypes = new ArrayList<>();
    statTypes.add("SampleCount");
    statTypes.add("Average");
    statTypes.add("Sum");
    statTypes.add("Minimum");
```

```
statTypes.add("Maximum");

for (int t = 0; t < 5; t++) {
    System.out.println("    " + (t + 1) + ". " + statTypes.get(t));
}
System.out.println("Select a metric statistic by entering a number from the
preceding list:");
num = Integer.parseInt(sc.nextLine());
if (1 <= num && num <= 5) {
    metricOption = statTypes.get(num - 1);
} else {
    System.out.println("You did not select a valid option.");
    System.exit(1);
}
System.out.println("You selected " + metricOption);
getAndDisplayMetricStatistics(cw, selectedNamespace, selectedMetrics,
metricOption, myDate, myDimension);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("4. Get CloudWatch estimated billing for the last
week.");
getMetricStatistics(cw, costDateWeek);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("5. Create a new CloudWatch dashboard with metrics.");
createDashboardWithMetrics(cw, dashboardName, dashboardJson);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("6. List dashboards using a paginator.");
listDashboards(cw);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("7. Create a new custom metric by adding data to it.");
createNewCustomMetric(cw, dataPoint);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("8. Add an additional metric to the dashboard.");
addMetricToDashboard(cw, dashboardAdd, dashboardName);
System.out.println(DASHES);
```

```
System.out.println(DASHES);
System.out.println("9. Create an alarm for the custom metric.");
String alarmName = createAlarm(cw, settings);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("10. Describe ten current alarms.");
describeAlarms(cw);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("11. Get current data for new custom metric.");
getCustomMetricData(cw, settings);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("12. Push data into the custom metric to trigger the
alarm.");
addMetricDataForAlarm(cw, settings);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("13. Check the alarm state using the action
DescribeAlarmsForMetric.");
checkForMetricAlarm(cw, settings);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("14. Get alarm history for the new alarm.");
getAlarmHistory(cw, settings, myDate);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("15. Add an anomaly detector for the custom metric.");
addAnomalyDetector(cw, settings);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("16. Describe current anomaly detectors.");
describeAnomalyDetectors(cw, settings);
System.out.println(DASHES);

System.out.println(DASHES);
```

```
System.out.println("17. Get a metric image for the custom metric.");
getAndOpenMetricImage(cw, metricImage);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("18. Clean up the Amazon CloudWatch resources.");
deleteDashboard(cw, dashboardName);
deleteCWAlarm(cw, alarmName);
deleteAnomalyDetector(cw, settings);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("The Amazon CloudWatch example scenario is complete.");
System.out.println(DASHES);
cw.close();
}

public static void deleteAnomalyDetector(CloudWatchClient cw, String fileName) {
    try {
        // Read values from the JSON file.
        JsonParser parser = new JsonFactory().createParser(new File(fileName));
        com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
        String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
        String customMetricName =
rootNode.findValue("customMetricName").asText();

        SingleMetricAnomalyDetector singleMetricAnomalyDetector =
SingleMetricAnomalyDetector.builder()
            .metricName(customMetricName)
            .namespace(customMetricNamespace)
            .stat("Maximum")
            .build();

        DeleteAnomalyDetectorRequest request =
DeleteAnomalyDetectorRequest.builder()
            .singleMetricAnomalyDetector(singleMetricAnomalyDetector)
            .build();

        cw.deleteAnomalyDetector(request);
        System.out.println("Successfully deleted the Anomaly Detector.");

    } catch (CloudWatchException e) {
```



```
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    } catch (IOException e) {
        e.printStackTrace();
    }
}

public static void deleteCWAlarm(CloudWatchClient cw, String alarmName) {
    try {
        DeleteAlarmsRequest request = DeleteAlarmsRequest.builder()
            .alarmNames(alarmName)
            .build();

        cw.deleteAlarms(request);
        System.out.println("Successfully deleted alarm " + alarmName);

    } catch (CloudWatchException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void deleteDashboard(CloudWatchClient cw, String dashboardName) {
    try {
        DeleteDashboardsRequest dashboardsRequest =
DeleteDashboardsRequest.builder()
            .dashboardNames(dashboardName)
            .build();
        cw.deleteDashboards(dashboardsRequest);
        System.out.println(dashboardName + " was successfully deleted.");

    } catch (CloudWatchException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void getAndOpenMetricImage(CloudWatchClient cw, String fileName) {
    System.out.println("Getting Image data for custom metric.");
    try {
        String myJSON = "{\n" +
            "  \"title\": \"Example Metric Graph\",\n" +
            "  \"view\": \"timeSeries\",\n" +
            "  \"stacked\": false,\n" +

```

```

        "  \"period\": 10,\n" +
        "  \"width\": 1400,\n" +
        "  \"height\": 600,\n" +
        "  \"metrics\": [\n" +
        "    [\n" +
        "      \"AWS/Billing\",\n" +
        "      \"EstimatedCharges\",\n" +
        "      \"Currency\",\n" +
        "      \"USD\"\n" +
        "    ]\n" +
        "  ]\n" +
        "];

    GetMetricWidgetImageRequest imageRequest =
    GetMetricWidgetImageRequest.builder()
        .metricWidget(myJSON)
        .build();

    GetMetricWidgetImageResponse response =
    cw.getMetricWidgetImage(imageRequest);
    SdkBytes sdkBytes = response.metricWidgetImage();
    byte[] bytes = sdkBytes.asByteArray();
    File outputFile = new File(fileName);
    try (FileOutputStream outputStream = new FileOutputStream(outputFile)) {
        outputStream.write(bytes);
    }

    } catch (CloudWatchException | IOException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void describeAnomalyDetectors(CloudWatchClient cw, String
fileName) {
    try {
        // Read values from the JSON file.
        JsonParser parser = new JsonFactory().createParser(new File(fileName));
        com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
        String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
        String customMetricName =
rootNode.findValue("customMetricName").asText();

```

```
DescribeAnomalyDetectorsRequest detectorsRequest =
DescribeAnomalyDetectorsRequest.builder()
    .maxResults(10)
    .metricName(customMetricName)
    .namespace(customMetricNamespace)
    .build();

DescribeAnomalyDetectorsResponse response =
cw.describeAnomalyDetectors(detectorsRequest);
List<AnomalyDetector> anomalyDetectorList = response.anomalyDetectors();
for (AnomalyDetector detector : anomalyDetectorList) {
    System.out.println("Metric name: " +
detector.singleMetricAnomalyDetector().metricName());
    System.out.println("State: " + detector.stateValue());
}

} catch (CloudWatchException | IOException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}

}

public static void addAnomalyDetector(CloudWatchClient cw, String fileName) {
    try {
        // Read values from the JSON file.
        JsonParser parser = new JsonFactory().createParser(new File(fileName));
        com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
        String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
        String customMetricName =
rootNode.findValue("customMetricName").asText();

        SingleMetricAnomalyDetector singleMetricAnomalyDetector =
SingleMetricAnomalyDetector.builder()
            .metricName(customMetricName)
            .namespace(customMetricNamespace)
            .stat("Maximum")
            .build();

        PutAnomalyDetectorRequest anomalyDetectorRequest =
PutAnomalyDetectorRequest.builder()
            .singleMetricAnomalyDetector(singleMetricAnomalyDetector)
            .build();
```

```
        cw.putAnomalyDetector(anomalyDetectorRequest);
        System.out.println("Added anomaly detector for metric " +
customMetricName + ".");

    } catch (CloudWatchException | IOException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void getAlarmHistory(CloudWatchClient cw, String fileName, String
date) {
    try {
        // Read values from the JSON file.
        JsonParser parser = new JsonFactory().createParser(new File(fileName));
        com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
        String alarmName = rootNode.findValue("exampleAlarmName").asText();

        Instant start = Instant.parse(date);
        Instant endDate = Instant.now();
        DescribeAlarmHistoryRequest historyRequest =
DescribeAlarmHistoryRequest.builder()
            .startDate(start)
            .endDate(endDate)
            .alarmName(alarmName)
            .historyItemType(HistoryItemType.ACTION)
            .build();

        DescribeAlarmHistoryResponse response =
cw.describeAlarmHistory(historyRequest);
        List<AlarmHistoryItem> historyItems = response.alarmHistoryItems();
        if (historyItems.isEmpty()) {
            System.out.println("No alarm history data found for " + alarmName +
".");
        } else {
            for (AlarmHistoryItem item : historyItems) {
                System.out.println("History summary: " + item.historySummary());
                System.out.println("Time stamp: " + item.timestamp());
            }
        }
    }

    } catch (CloudWatchException | IOException e) {
```

```
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void checkForMetricAlarm(CloudWatchClient cw, String fileName) {
    try {
        // Read values from the JSON file.
        JsonParser parser = new JsonFactory().createParser(new File(fileName));
        com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
        String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
        String customMetricName =
rootNode.findValue("customMetricName").asText();
        boolean hasAlarm = false;
        int retries = 10;

        DescribeAlarmsForMetricRequest metricRequest =
DescribeAlarmsForMetricRequest.builder()
            .metricName(customMetricName)
            .namespace(customMetricNamespace)
            .build();

        while (!hasAlarm && retries > 0) {
            DescribeAlarmsForMetricResponse response =
cw.describeAlarmsForMetric(metricRequest);
            hasAlarm = response.hasMetricAlarms();
            retries--;
            Thread.sleep(20000);
            System.out.println(".");
        }
        if (!hasAlarm)
            System.out.println("No Alarm state found for " + customMetricName +
" after 10 retries.");
        else
            System.out.println("Alarm state found for " + customMetricName +
".");

    } catch (CloudWatchException | IOException | InterruptedException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

```
public static void addMetricDataForAlarm(CloudWatchClient cw, String fileName) {
    try {
        // Read values from the JSON file.
        JsonParser parser = new JsonFactory().createParser(new File(fileName));
        com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
        String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
        String customMetricName =
rootNode.findValue("customMetricName").asText();

        // Set an Instant object.
        String time =
ZonedDateTime.now(ZoneOffset.UTC).format(DateTimeFormatter.ISO_INSTANT);
        Instant instant = Instant.parse(time);

        MetricDatum datum = MetricDatum.builder()
            .metricName(customMetricName)
            .unit(StandardUnit.NONE)
            .value(1001.00)
            .timestamp(instant)
            .build();

        MetricDatum datum2 = MetricDatum.builder()
            .metricName(customMetricName)
            .unit(StandardUnit.NONE)
            .value(1002.00)
            .timestamp(instant)
            .build();

        List<MetricDatum> metricDataList = new ArrayList<>();
        metricDataList.add(datum);
        metricDataList.add(datum2);

        PutMetricDataRequest request = PutMetricDataRequest.builder()
            .namespace(customMetricNamespace)
            .metricData(metricDataList)
            .build();

        cw.putMetricData(request);
        System.out.println("Added metric values for for metric " +
customMetricName);
    }
}
```

```
    } catch (CloudWatchException | IOException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void getCustomMetricData(CloudWatchClient cw, String fileName) {
    try {
        // Read values from the JSON file.
        JsonParser parser = new JsonFactory().createParser(new File(fileName));
        com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
        String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
        String customMetricName =
rootNode.findValue("customMetricName").asText();

        // Set the date.
        Instant nowDate = Instant.now();

        long hours = 1;
        long minutes = 30;
        Instant date2 = nowDate.plus(hours, ChronoUnit.HOURS).plus(minutes,
ChronoUnit.MINUTES);

        Metric met = Metric.builder()
            .metricName(customMetricName)
            .namespace(customMetricNamespace)
            .build();

        MetricStat metStat = MetricStat.builder()
            .stat("Maximum")
            .period(1)
            .metric(met)
            .build();

        MetricDataQuery dataQuery = MetricDataQuery.builder()
            .metricStat(metStat)
            .id("foo2")
            .returnData(true)
            .build();

        List<MetricDataQuery> dq = new ArrayList<>();
        dq.add(dataQuery);
    }
}
```

```
GetMetricDataRequest getMetReq = GetMetricDataRequest.builder()
    .maxDatapoints(10)
    .scanBy(ScanBy.TIMESTAMP_DESCENDING)
    .startTime(nowDate)
    .endTime(date2)
    .metricDataQueries(dq)
    .build();

GetMetricDataResponse response = cw.getMetricData(getMetReq);
List<MetricDataResult> data = response.metricDataResults();
for (MetricDataResult item : data) {
    System.out.println("The label is " + item.label());
    System.out.println("The status code is " +
item.statusCode().toString());
}

} catch (CloudWatchException | IOException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
}

public static void describeAlarms(CloudWatchClient cw) {
    try {
        List<AlarmType> typeList = new ArrayList<>();
        typeList.add(AlarmType.METRIC_ALARM);

        DescribeAlarmsRequest alarmsRequest = DescribeAlarmsRequest.builder()
            .alarmTypes(typeList)
            .maxRecords(10)
            .build();

        DescribeAlarmsResponse response = cw.describeAlarms(alarmsRequest);
        List<MetricAlarm> alarmList = response.metricAlarms();
        for (MetricAlarm alarm : alarmList) {
            System.out.println("Alarm name: " + alarm.alarmName());
            System.out.println("Alarm description: " +
alarm.alarmDescription());
        }
    } catch (CloudWatchException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```



```
}

public static String createAlarm(CloudWatchClient cw, String fileName) {
    try {
        // Read values from the JSON file.
        JsonParser parser = new JsonFactory().createParser(new File(fileName));
        com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
        String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
        String customMetricName =
rootNode.findValue("customMetricName").asText();
        String alarmName = rootNode.findValue("exampleAlarmName").asText();
        String emailTopic = rootNode.findValue("emailTopic").asText();
        String accountId = rootNode.findValue("accountId").asText();
        String region = rootNode.findValue("region").asText();

        // Create a List for alarm actions.
        List<String> alarmActions = new ArrayList<>();
        alarmActions.add("arn:aws:sns:" + region + ":" + accountId + ":" +
emailTopic);
        PutMetricAlarmRequest alarmRequest = PutMetricAlarmRequest.builder()
            .alarmActions(alarmActions)
            .alarmDescription("Example metric alarm")
            .alarmName(alarmName)

.comparisonOperator(ComparisonOperator.GREATER_THAN_OR_EQUAL_TO_THRESHOLD)
            .threshold(100.00)
            .metricName(customMetricName)
            .namespace(customMetricNamespace)
            .evaluationPeriods(1)
            .period(10)
            .statistic("Maximum")
            .datapointsToAlarm(1)
            .treatMissingData("ignore")
            .build();

        cw.putMetricAlarm(alarmRequest);
        System.out.println(alarmName + " was successfully created!");
        return alarmName;

    } catch (CloudWatchException | IOException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

```
    }
    return "";
}

public static void addMetricToDashboard(CloudWatchClient cw, String fileName,
String dashboardName) {
    try {
        PutDashboardRequest dashboardRequest = PutDashboardRequest.builder()
            .dashboardName(dashboardName)
            .dashboardBody(readFileAsString(fileName))
            .build();

        cw.putDashboard(dashboardRequest);
        System.out.println(dashboardName + " was successfully updated.");

    } catch (CloudWatchException | IOException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void createNewCustomMetric(CloudWatchClient cw, Double dataPoint)
{
    try {
        Dimension dimension = Dimension.builder()
            .name("UNIQUE_PAGES")
            .value("URLS")
            .build();

        // Set an Instant object.
        String time =
ZonedDateTime.now(ZoneOffset.UTC).format(DateTimeFormatter.ISO_INSTANT);
        Instant instant = Instant.parse(time);

        MetricDatum datum = MetricDatum.builder()
            .metricName("PAGES_VISITED")
            .unit(StandardUnit.NONE)
            .value(dataPoint)
            .timestamp(instant)
            .dimensions(dimension)
            .build();

        PutMetricDataRequest request = PutMetricDataRequest.builder()
            .namespace("SITE/TRAFFIC")
```

```

        .metricData(datum)
        .build();

        cw.putMetricData(request);
        System.out.println("Added metric values for for metric PAGES_VISITED");

    } catch (CloudWatchException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void listDashboards(CloudWatchClient cw) {
    try {
        ListDashboardsIterable listRes = cw.listDashboardsPaginator();
        listRes.stream()
            .flatMap(r -> r.dashboardEntries().stream())
            .forEach(entry -> {
                System.out.println("Dashboard name is: " +
entry.dashboardName());
                System.out.println("Dashboard ARN is: " +
entry.dashboardArn());
            });

    } catch (CloudWatchException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void createDashboardWithMetrics(CloudWatchClient cw, String
dashboardName, String fileName) {
    try {
        PutDashboardRequest dashboardRequest = PutDashboardRequest.builder()
            .dashboardName(dashboardName)
            .dashboardBody(readFileAsString(fileName))
            .build();

        PutDashboardResponse response = cw.putDashboard(dashboardRequest);
        System.out.println(dashboardName + " was successfully created.");
        List<DashboardValidationMessage> messages =
response.dashboardValidationMessages();
        if (messages.isEmpty()) {
            System.out.println("There are no messages in the new Dashboard");
        }
    }
}

```

```
        } else {
            for (DashboardValidationMessage message : messages) {
                System.out.println("Message is: " + message.message());
            }
        }

    } catch (CloudWatchException | IOException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static String readFileAsString(String file) throws IOException {
    return new String(Files.readAllBytes(Paths.get(file)));
}

public static void getMetricStatistics(CloudWatchClient cw, String costDateWeek)
{
    try {
        Instant start = Instant.parse(costDateWeek);
        Instant endDate = Instant.now();
        Dimension dimension = Dimension.builder()
            .name("Currency")
            .value("USD")
            .build();

        List<Dimension> dimensionList = new ArrayList<>();
        dimensionList.add(dimension);
        GetMetricStatisticsRequest statisticsRequest =
        GetMetricStatisticsRequest.builder()
            .metricName("EstimatedCharges")
            .namespace("AWS/Billing")
            .dimensions(dimensionList)
            .statistics(Statistic.MAXIMUM)
            .startTime(start)
            .endTime(endDate)
            .period(86400)
            .build();

        GetMetricStatisticsResponse response =
        cw.getMetricStatistics(statisticsRequest);
        List<Datapoint> data = response.datapoints();
        if (!data.isEmpty()) {
            for (Datapoint datapoint : data) {
```

```

        System.out
            .println("Timestamp: " + datapoint.timestamp() + "
Maximum value: " + datapoint.maximum());
    }
    } else {
        System.out.println("The returned data list is empty");
    }

} catch (CloudWatchException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}

public static void getAndDisplayMetricStatistics(CloudWatchClient cw, String
nameSpace, String metVal,
String metricOption, String date, Dimension myDimension) {
    try {
        Instant start = Instant.parse(date);
        Instant endDate = Instant.now();

        GetMetricStatisticsRequest statisticsRequest =
GetMetricStatisticsRequest.builder()
            .endTime(endDate)
            .startTime(start)
            .dimensions(myDimension)
            .metricName(metVal)
            .namespace(nameSpace)
            .period(86400)
            .statistics(Statistic.fromValue(metricOption))
            .build();

        GetMetricStatisticsResponse response =
cw.getMetricStatistics(statisticsRequest);
        List<Datapoint> data = response.datapoints();
        if (!data.isEmpty()) {
            for (Datapoint datapoint : data) {
                System.out
                    .println("Timestamp: " + datapoint.timestamp() + "
Maximum value: " + datapoint.maximum());
            }
        } else {
            System.out.println("The returned data list is empty");
        }
    }
}

```

```
    } catch (CloudWatchException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static Dimension getSpecificMet(CloudWatchClient cw, String namespace) {
    try {
        ListMetricsRequest request = ListMetricsRequest.builder()
            .namespace(namespace)
            .build();

        ListMetricsResponse response = cw.listMetrics(request);
        List<Metric> myList = response.metrics();
        Metric metric = myList.get(0);
        return metric.dimensions().get(0);

    } catch (CloudWatchException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return null;
}

public static ArrayList<String> listMets(CloudWatchClient cw, String namespace)
{
    try {
        ArrayList<String> metList = new ArrayList<>();
        ListMetricsRequest request = ListMetricsRequest.builder()
            .namespace(namespace)
            .build();

        ListMetricsIterable listRes = cw.listMetricsPaginator(request);
        listRes.stream()
            .flatMap(r -> r.metrics().stream())
            .forEach(metrics -> metList.add(metrics.metricName()));

        return metList;

    } catch (CloudWatchException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

```
        return null;
    }

    public static ArrayList<String> listNameSpaces(CloudWatchClient cw) {
        try {
            ArrayList<String> nameSpaceList = new ArrayList<>();
            ListMetricsRequest request = ListMetricsRequest.builder()
                .build();

            ListMetricsIterable listRes = cw.listMetricsPaginator(request);
            listRes.stream()
                .flatMap(r -> r.metrics().stream())
                .forEach(metrics -> {
                    String data = metrics.namespace();
                    if (!nameSpaceList.contains(data)) {
                        nameSpaceList.add(data);
                    }
                });

            return nameSpaceList;
        } catch (CloudWatchException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return null;
    }
}
```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella Documentazione di riferimento delle API AWS SDK for Java 2.x .
  - [DeleteAlarms](#)
  - [DeleteAnomalyDetector](#)
  - [DeleteDashboards](#)
  - [DescribeAlarmHistory](#)
  - [DescribeAlarms](#)
  - [DescribeAlarmsForMetric](#)
  - [DescribeAnomalyDetectors](#)
  - [GetMetricData](#)

- [GetMetricStatistics](#)
- [GetMetricWidgetImage](#)
- [ListMetrics](#)
- [PutAnomalyDetector](#)
- [PutDashboard](#)
- [PutMetricAlarm](#)
- [PutMetricData](#)

## CloudWatch Esempi di eventi che utilizzano SDK for Java 2.x

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando AWS SDK for Java 2.x with CloudWatch Events.

Le operazioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le operazioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati e negli esempi tra servizi.

Scenari: esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio.

Ogni esempio include un collegamento a GitHub, dove è possibile trovare istruzioni su come configurare ed eseguire il codice nel contesto.

Argomenti

- [Azioni](#)

Azioni

### **PutEvents**

Il seguente esempio di codice mostra come utilizzarePutEvents.



## SDK per Java 2.x

 Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatchevents.CloudWatchEventsClient;
import software.amazon.awssdk.services.cloudwatchevents.model.PutEventsRequest;
import software.amazon.awssdk.services.cloudwatchevents.model.PutEventsRequestEntry;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class PutEvents {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <resourceArn>

                Where:
                resourceArn - An Amazon Resource Name (ARN) related to the
events.

                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String resourceArn = args[0];
        CloudWatchEventsClient cwe = CloudWatchEventsClient.builder()
                .build();
```

```
        putCWEEvents(cwe, resourceArn);
        cwe.close();
    }

    public static void putCWEEvents(CloudWatchEventsClient cwe, String resourceArn) {
        try {
            final String EVENT_DETAILS = "{ \"key1\": \"value1\", \"key2\": \"value2\" }";

            PutEventsRequestEntry requestEntry = PutEventsRequestEntry.builder()
                .detail(EVENT_DETAILS)
                .detailType("sampleSubmitted")
                .resources(resourceArn)
                .source("aws-sdk-java-cloudwatch-example")
                .build();

            PutEventsRequest request = PutEventsRequest.builder()
                .entries(requestEntry)
                .build();

            cwe.putEvents(request);
            System.out.println("Successfully put CloudWatch event");

        } catch (CloudWatchException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Per i dettagli sull'API, consulta la [PutEvents](#) sezione AWS SDK for Java 2.x API Reference.

## PutRule

Il seguente esempio di codice mostra come utilizzare `PutRule`.

## SDK per Java 2.x

 Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatchevents.CloudWatchEventsClient;
import software.amazon.awssdk.services.cloudwatchevents.model.PutRuleRequest;
import software.amazon.awssdk.services.cloudwatchevents.model.PutRuleResponse;
import software.amazon.awssdk.services.cloudwatchevents.model.RuleState;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class PutRule {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <ruleName> roleArn\s

            Where:
                ruleName - A rule name (for example, myrule).
                roleArn - A role ARN value (for example,
arn:aws:iam::xxxxxx047983:user/MyUser).
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String ruleName = args[0];
        String roleArn = args[1];
```

```
CloudWatchEventsClient cwe = CloudWatchEventsClient.builder()
    .build();

putCWRule(cwe, ruleName, roleArn);
cwe.close();
}

public static void putCWRule(CloudWatchEventsClient cwe, String ruleName, String
roleArn) {
    try {
        PutRuleRequest request = PutRuleRequest.builder()
            .name(ruleName)
            .roleArn(roleArn)
            .scheduleExpression("rate(5 minutes)")
            .state(RuleState.ENABLED)
            .build();

        PutRuleResponse response = cwe.putRule(request);
        System.out.printf(
            "Successfully created CloudWatch events rule %s with arn %s",
            ruleArn, response.ruleArn());

    } catch (CloudWatchException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Per i dettagli sull'API, consulta la [PutRule](#) sezione AWS SDK for Java 2.x API Reference.

## PutTargets

Il seguente esempio di codice mostra come utilizzare PutTargets.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatchevents.CloudWatchEventsClient;
import software.amazon.awssdk.services.cloudwatchevents.model.PutTargetsRequest;
import software.amazon.awssdk.services.cloudwatchevents.model.Target;

/**
 * To run this Java V2 code example, ensure that you have setup your development
 * environment, including your credentials.
 *
 * For information, see this documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class PutTargets {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <ruleName> <functionArn> <targetId>\s

                Where:
                ruleName - A rule name (for example, myrule).
                functionArn - An AWS Lambda function ARN (for example,
                arn:aws:lambda:us-west-2:xxxxxx047983:function:lamda1).
                targetId - A target id value.
                """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String ruleName = args[0];
        String functionArn = args[1];
        String targetId = args[2];
        CloudWatchEventsClient cwe = CloudWatchEventsClient.builder()
                .build();

        putCWTargets(cwe, ruleName, functionArn, targetId);
        cwe.close();
    }
}
```

```
public static void putCWTargets(CloudWatchEventsClient cwe, String ruleName,
String functionArn, String targetId) {
    try {
        Target target = Target.builder()
            .arn(functionArn)
            .id(targetId)
            .build();

        PutTargetsRequest request = PutTargetsRequest.builder()
            .targets(target)
            .rule(ruleName)
            .build();

        cwe.putTargets(request);
        System.out.printf(
            "Successfully created CloudWatch events target for rule %s",
            ruleName);

    } catch (CloudWatchException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Per i dettagli sull'API, consulta la [PutTargets](#) sezione AWS SDK for Java 2.x API Reference.

## CloudWatch Esempi di log utilizzando SDK for Java 2.x

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando AWS SDK for Java 2.x with CloudWatch Logs.

Le operazioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le operazioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati e negli esempi tra servizi.

Scenari: esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio.

Ogni esempio include un collegamento a GitHub, dove è possibile trovare istruzioni su come configurare ed eseguire il codice nel contesto.

## Argomenti

- [Azioni](#)

## Azioni

### DeleteSubscriptionFilter

Il seguente esempio di codice mostra come utilizzare DeleteSubscriptionFilter.

SDK per Java 2.x

#### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatchlogs.CloudWatchLogsClient;
import
    software.amazon.awssdk.services.cloudwatchlogs.model.DeleteSubscriptionFilterRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteSubscriptionFilter {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <filter> <logGroup>

                Where:
                filter - The name of the subscription filter (for example,
MyFilter).
                logGroup - The name of the log group. (for example, testgroup).
```

```
        """);

    if (args.length != 2) {
        System.out.println(usage);
        System.exit(1);
    }

    String filter = args[0];
    String logGroup = args[1];
    CloudWatchLogsClient logs = CloudWatchLogsClient.builder()
        .build();

    deleteSubFilter(logs, filter, logGroup);
    logs.close();
}

public static void deleteSubFilter(CloudWatchLogsClient logs, String filter,
String logGroup) {
    try {
        DeleteSubscriptionFilterRequest request =
DeleteSubscriptionFilterRequest.builder()
            .filterName(filter)
            .logGroupName(logGroup)
            .build();

        logs.deleteSubscriptionFilter(request);
        System.out.printf("Successfully deleted CloudWatch logs subscription
filter %s", filter);

    } catch (CloudWatchException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Per i dettagli sull'API, consulta la [DeleteSubscriptionFilter](#) sezione AWS SDK for Java 2.x API Reference.

## DescribeSubscriptionFilters

Il seguente esempio di codice mostra come utilizzare `DescribeSubscriptionFilters`.



## SDK per Java 2.x

 Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatchlogs.CloudWatchLogsClient;
import
    software.amazon.awssdk.services.cloudwatchlogs.model.DescribeSubscriptionFiltersRequest;
import
    software.amazon.awssdk.services.cloudwatchlogs.model.DescribeSubscriptionFiltersResponse;
import software.amazon.awssdk.services.cloudwatchlogs.model.SubscriptionFilter;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DescribeSubscriptionFilters {
    public static void main(String[] args) {

        final String usage = ""

            Usage:
            <logGroup>

            Where:
            logGroup - A log group name (for example, myloggroup).
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }
    }
}
```

```
String logGroup = args[0];
CloudWatchLogsClient logs = CloudWatchLogsClient.builder()
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

describeFilters(logs, logGroup);
logs.close();
}

public static void describeFilters(CloudWatchLogsClient logs, String logGroup) {
    try {
        boolean done = false;
        String newToken = null;

        while (!done) {
            DescribeSubscriptionFiltersResponse response;
            if (newToken == null) {
                DescribeSubscriptionFiltersRequest request =
DescribeSubscriptionFiltersRequest.builder()
                    .logGroupName(logGroup)
                    .limit(1).build();

                response = logs.describeSubscriptionFilters(request);
            } else {
                DescribeSubscriptionFiltersRequest request =
DescribeSubscriptionFiltersRequest.builder()
                    .nextToken(newToken)
                    .logGroupName(logGroup)
                    .limit(1).build();
                response = logs.describeSubscriptionFilters(request);
            }

            for (SubscriptionFilter filter : response.subscriptionFilters()) {
                System.out.printf("Retrieved filter with name %s, " + "pattern
%s " + "and destination arn %s",
                    filter.filterName(),
                    filter.filterPattern(),
                    filter.destinationArn());
            }

            if (response.nextToken() == null) {
                done = true;
            } else {
                newToken = response.nextToken();
            }
        }
    }
}
```

```

        }
    }

    } catch (CloudWatchException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    System.out.printf("Done");
}
}

```

- Per i dettagli sull'API, consulta la [DescribeSubscriptionFilters](#) sezione AWS SDK for Java 2.x API Reference.

## PutSubscriptionFilter

Il seguente esempio di codice mostra come utilizzare `PutSubscriptionFilter`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudwatchlogs.CloudWatchLogsClient;
import software.amazon.awssdk.services.cloudwatchlogs.model.CloudWatchLogsException;
import
    software.amazon.awssdk.services.cloudwatchlogs.model.PutSubscriptionFilterRequest;

/**
 * Before running this code example, you need to grant permission to CloudWatch
 * Logs the right to execute your Lambda function.
 * To perform this task, you can use this CLI command:
 *
 * aws lambda add-permission --function-name "lamda1" --statement-id "lamda1"
 * --principal "logs.us-west-2.amazonaws.com" --action "lambda:InvokeFunction"
 * --source-arn "arn:aws:logs:us-west-2:111111111111:log-group:testgroup:*"

```

```

* --source-account "111111111111"
*
* Make sure you replace the function name with your function name and replace
* '111111111111' with your account details.
* For more information, see "Subscription Filters with AWS Lambda" in the
* Amazon CloudWatch Logs Guide.
*
*
* Also, before running this Java V2 code example, set up your development
* environment, including your credentials.
*
* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*
*/

```

```

public class PutSubscriptionFilter {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <filter> <pattern> <logGroup> <functionArn>\s

            Where:
                filter - A filter name (for example, myfilter).
                pattern - A filter pattern (for example, ERROR).
                logGroup - A log group name (testgroup).
                functionArn - An AWS Lambda function ARN (for example,
arn:aws:lambda:us-west-2:111111111111:function:lambda1) .
                """;

        if (args.length != 4) {
            System.out.println(usage);
            System.exit(1);
        }

        String filter = args[0];
        String pattern = args[1];
        String logGroup = args[2];
        String functionArn = args[3];
        Region region = Region.US_WEST_2;
        CloudWatchLogsClient cw1 = CloudWatchLogsClient.builder()
            .region(region)

```

```
        .build();

        putSubFilters(cwl, filter, pattern, logGroup, functionArn);
        cwl.close();
    }

    public static void putSubFilters(CloudWatchLogsClient cwl,
        String filter,
        String pattern,
        String logGroup,
        String functionArn) {

        try {
            PutSubscriptionFilterRequest request =
PutSubscriptionFilterRequest.builder()
                .filterName(filter)
                .filterPattern(pattern)
                .logGroupName(logGroup)
                .destinationArn(functionArn)
                .build();

            cwl.putSubscriptionFilter(request);
            System.out.printf(
                "Successfully created CloudWatch logs subscription filter %s",
                filter);

        } catch (CloudWatchLogsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Per i dettagli sull'API, consulta la [PutSubscriptionFilter](#) sezione AWS SDK for Java 2.x API Reference.

## StartLiveTail

Il seguente esempio di codice mostra come utilizzare `StartLiveTail`.

## SDK per Java 2.x

Includere i file richiesti.

```
import io.reactivex.FlowableSubscriber;
import io.reactivex.annotations.NonNull;
import org.reactivestreams.Subscription;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.services.cloudwatchlogs.CloudWatchLogsAsyncClient;
import software.amazon.awssdk.services.cloudwatchlogs.model.LiveTailSessionLogEvent;
import software.amazon.awssdk.services.cloudwatchlogs.model.LiveTailSessionStart;
import software.amazon.awssdk.services.cloudwatchlogs.model.LiveTailSessionUpdate;
import software.amazon.awssdk.services.cloudwatchlogs.model.StartLiveTailRequest;
import
    software.amazon.awssdk.services.cloudwatchlogs.model.StartLiveTailResponseHandler;
import software.amazon.awssdk.services.cloudwatchlogs.model.CloudWatchLogsException;
import
    software.amazon.awssdk.services.cloudwatchlogs.model.StartLiveTailResponseStream;

import java.util.Date;
import java.util.List;
import java.util.concurrent.atomic.AtomicReference;
```

Gestisci gli eventi della sessione Live Tail.

```
private static StartLiveTailResponseHandler
getStartLiveTailResponseStreamHandler(
    AtomicReference<Subscription> subscriptionAtomicReference) {
    return StartLiveTailResponseHandler.builder()
        .onResponse(r -> System.out.println("Received initial response"))
        .onError(throwable -> {
            CloudWatchLogsException e = (CloudWatchLogsException)
throwable.getCause();
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        })
        .subscriber(() -> new FlowableSubscriber<>() {
            @Override
            public void onSubscribe(@NonNull Subscription s) {
                subscriptionAtomicReference.set(s);
                s.request(Long.MAX_VALUE);
            }
        })
    }
```

```

        @Override
        public void onNext(StartLiveTailResponseStream event) {
            if (event instanceof LiveTailSessionStart) {
                LiveTailSessionStart sessionStart = (LiveTailSessionStart)
event;

                System.out.println(sessionStart);
            } else if (event instanceof LiveTailSessionUpdate) {
                LiveTailSessionUpdate sessionUpdate =
(LiveTailSessionUpdate) event;
                List<LiveTailSessionLogEvent> logEvents =
sessionUpdate.sessionResults();
                logEvents.forEach(e -> {
                    long timestamp = e.timestamp();
                    Date date = new Date(timestamp);
                    System.out.println "[" + date + "] " + e.message());
                });
            } else {
                throw CloudWatchLogsException.builder().message("Unknown
event type").build();
            }
        }

        @Override
        public void onError(Throwable throwable) {
            System.out.println(throwable.getMessage());
            System.exit(1);
        }

        @Override
        public void onComplete() {
            System.out.println("Completed Streaming Session");
        }
    })
    .build();
}

```

Avvia la sessione Live Tail.

```

CloudWatchLogsAsyncClient cloudWatchLogsAsyncClient =
    CloudWatchLogsAsyncClient.builder()
        .credentialsProvider(ProfileCredentialsProvider.create())

```

```

        .build();

    StartLiveTailRequest request =
        StartLiveTailRequest.builder()
            .logGroupIdentifiers(logGroupIdentifiers)
            .logStreamNames(logStreamNames)
            .logEventFilterPattern(logEventFilterPattern)
            .build();

    /* Create a reference to store the subscription */
    final AtomicReference<Subscription> subscriptionAtomicReference = new
AtomicReference<>(null);

    cloudWatchLogsAsyncClient.startLiveTail(request,
getStartLiveTailResponseStreamHandler(subscriptionAtomicReference));

```

Interrompi la sessione Live Tail dopo un certo periodo di tempo.

```

/* Set a timeout for the session and cancel the subscription. This will:
 * 1). Close the stream
 * 2). Stop the Live Tail session
 */
try {
    Thread.sleep(10000);
} catch (InterruptedException e) {
    throw new RuntimeException(e);
}
if (subscriptionAtomicReference.get() != null) {
    subscriptionAtomicReference.get().cancel();
    System.out.println("Subscription to stream closed");
}

```

- Per i dettagli sulle API, consulta la sezione AWS SDK for Java 2.x API [StartLiveTailReference](#).

## Esempi di identità di Amazon Cognito con SDK for Java 2.x

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando Amazon Cognito Identity. AWS SDK for Java 2.x



Le operazioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le operazioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati e negli esempi tra servizi.

Scenari: esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio.

Ogni esempio include un collegamento a GitHub, dove puoi trovare istruzioni su come configurare ed eseguire il codice nel contesto.

Argomenti

- [Azioni](#)

Azioni

## CreateIdentityPool

Il seguente esempio di codice mostra come utilizzare `CreateIdentityPool`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cognitoidentity.CognitoIdentityClient;
import
    software.amazon.awssdk.services.cognitoidentity.model.CreateIdentityPoolRequest;
import
    software.amazon.awssdk.services.cognitoidentity.model.CreateIdentityPoolResponse;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
```

```
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class CreateIdentityPool {
    public static void main(String[] args) {
        final String usage = ""
            Usage:
            <identityPoolName>\s

            Where:
            identityPoolName - The name to give your identity pool.
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String identityPoolName = args[0];
        CognitoIdentityClient cognitoClient = CognitoIdentityClient.builder()
            .region(Region.US_EAST_1)
            .build();

        String identityPoolId = createIdPool(cognitoClient, identityPoolName);
        System.out.println("Unity pool ID " + identityPoolId);
        cognitoClient.close();
    }

    public static String createIdPool(CognitoIdentityClient cognitoClient, String
identityPoolName) {
        try {
            CreateIdentityPoolRequest poolRequest =
CreateIdentityPoolRequest.builder()
                .allowUnauthenticatedIdentities(false)
                .identityPoolName(identityPoolName)
                .build();

            CreateIdentityPoolResponse response =
cognitoClient.createIdentityPool(poolRequest);
            return response.identityPoolId();
        } catch (CognitoIdentityProviderException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

```
    }  
    return "";  
  }  
}
```

- Per i dettagli sull'API, consulta la [CreateIdentityPool](#) sezione AWS SDK for Java 2.x API Reference.

## DeleteIdentityPool

Il seguente esempio di codice mostra come utilizzare `DeleteIdentityPool`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;  
import software.amazon.awssdk.awscore.exception.AwsServiceException;  
import software.amazon.awssdk.regions.Region;  
import software.amazon.awssdk.services.cognitoidentity.CognitoIdentityClient;  
import  
    software.amazon.awssdk.services.cognitoidentity.model.DeleteIdentityPoolRequest;  
  
/**  
 * Before running this Java V2 code example, set up your development  
 * environment, including your credentials.  
 *  
 * For more information, see the following documentation topic:  
 *  
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html  
 */  
public class DeleteIdentityPool {  
  
    public static void main(String[] args) {  
        final String usage = ""
```

Usage:

```
<identityPoolId>\s
```

Where:

```
identityPoolId - The Id value of your identity pool.  
"";
```

```
if (args.length != 1) {  
    System.out.println(usage);  
    System.exit(1);  
}  
  
String identityPoolId = args[0];  
CognitoIdentityClient cognitoIdClient = CognitoIdentityClient.builder()  
    .region(Region.US_EAST_1)  
    .credentialsProvider(ProfileCredentialsProvider.create())  
    .build();  
  
deleteIdPool(cognitoIdClient, identityPoolId);  
cognitoIdClient.close();  
}  
  
public static void deleteIdPool(CognitoIdentityClient cognitoIdClient, String  
identityPoolId) {  
    try {  
  
        DeleteIdentityPoolRequest identityPoolRequest =  
DeleteIdentityPoolRequest.builder()  
            .identityPoolId(identityPoolId)  
            .build();  
  
        cognitoIdClient.deleteIdentityPool(identityPoolRequest);  
        System.out.println("Done");  
  
    } catch (AwsServiceException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}  
}
```

- Per i dettagli sull'API, consulta la [DeleteIdentityPool](#) sezione AWS SDK for Java 2.x API Reference.

## GetCredentialsForIdentity

Il seguente esempio di codice mostra come utilizzare `GetCredentialsForIdentity`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cognitoidentity.CognitoIdentityClient;
import
    software.amazon.awssdk.services.cognitoidentity.model.GetCredentialsForIdentityRequest;
import
    software.amazon.awssdk.services.cognitoidentity.model.GetCredentialsForIdentityResponse;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class GetIdentityCredentials {
    public static void main(String[] args) {

        final String usage = ""

            Usage:
                <identityId>\s

        Where:
```

```

        identityId - The Id of an existing identity in the format
REGION:GUID.
        """;

    if (args.length != 1) {
        System.out.println(usage);
        System.exit(1);
    }

    String identityId = args[0];
    CognitoIdentityClient cognitoClient = CognitoIdentityClient.builder()
        .region(Region.US_EAST_1)
        .build();

    getCredsForIdentity(cognitoClient, identityId);
    cognitoClient.close();
}

public static void getCredsForIdentity(CognitoIdentityClient cognitoClient,
String identityId) {
    try {
        GetCredentialsForIdentityRequest getCredentialsForIdentityRequest =
GetCredentialsForIdentityRequest
        .builder()
        .identityId(identityId)
        .build();

        GetCredentialsForIdentityResponse response = cognitoClient
        .getCredentialsForIdentity(getCredentialsForIdentityRequest);
        System.out.println(
            "Identity ID " + response.identityId() + ", Access key ID " +
response.credentials().accessKeyId());

    } catch (CognitoIdentityProviderException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
}

```

- Per i dettagli sull'API, consulta la [GetCredentialsForIdentity](#) sezione AWS SDK for Java 2.x API Reference.

## ListIdentityPools

Il seguente esempio di codice mostra come utilizzare `ListIdentityPools`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cognitoidentity.CognitoIdentityClient;
import
    software.amazon.awssdk.services.cognitoidentity.model.ListIdentityPoolsRequest;
import
    software.amazon.awssdk.services.cognitoidentity.model.ListIdentityPoolsResponse;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListIdentityPools {
    public static void main(String[] args) {
        CognitoIdentityClient cognitoClient = CognitoIdentityClient.builder()
            .region(Region.US_EAST_1)
            .build();

        listIdPools(cognitoClient);
        cognitoClient.close();
    }

    public static void listIdPools(CognitoIdentityClient cognitoClient) {
        try {
            ListIdentityPoolsRequest poolsRequest =
                ListIdentityPoolsRequest.builder()
```

```
        .maxResults(15)
        .build();

    ListIdentityPoolsResponse response =
cognitoClient.listIdentityPools(poolsRequest);
    response.identityPools().forEach(pool -> {
        System.out.println("Pool ID: " + pool.identityPoolId());
        System.out.println("Pool name: " + pool.identityPoolName());
    });

    } catch (CognitoIdentityProviderException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Per i dettagli sull'API, consulta la [ListIdentityPools](#) sezione AWS SDK for Java 2.x API Reference.

## Esempi di Amazon Cognito Identity Provider che utilizzano SDK for Java 2.x

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando Amazon Cognito Identity Provider. AWS SDK for Java 2.x

Le operazioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le operazioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati e negli esempi tra servizi.

Scenari: esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio.

Ogni esempio include un collegamento a GitHub, dove puoi trovare istruzioni su come configurare ed eseguire il codice nel contesto.

Nozioni di base

Ciao Amazon Cognito

Gli esempi di codice seguente mostrano come iniziare a utilizzare Amazon Cognito.



## SDK per Java 2.x

 Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;
import
    software.amazon.awssdk.services.cognitoidentityprovider.CognitoIdentityProviderClient;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderException;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.ListUserPoolsResponse;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.ListUserPoolsRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListUserPools {
    public static void main(String[] args) {
        CognitoIdentityProviderClient cognitoClient =
        CognitoIdentityProviderClient.builder()
            .region(Region.US_EAST_1)
            .build();

        listAllUserPools(cognitoClient);
        cognitoClient.close();
    }

    public static void listAllUserPools(CognitoIdentityProviderClient cognitoClient)
    {
        try {
            ListUserPoolsRequest request = ListUserPoolsRequest.builder()
                .maxResults(10)
```

```
        .build();

        ListUserPoolsResponse response = cognitoClient.listUserPools(request);
        response.userPools().forEach(userpool -> {
            System.out.println("User pool " + userpool.name() + ", User ID " +
                userpool.id());
        });

        } catch (CognitoIdentityProviderException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Per i dettagli sull'API, consulta la [ListUserPools](#) sezione AWS SDK for Java 2.x API Reference.

## Argomenti

- [Azioni](#)
- [Scenari](#)

## Azioni

### AdminGetUser

Il seguente esempio di codice mostra come utilizzare `AdminGetUser`.

SDK per Java 2.x

#### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public static void getAdminUser(CognitoIdentityProviderClient
    identityProviderClient, String userName,
        String poolId) {
```

```
try {
    AdminGetUserRequest userRequest = AdminGetUserRequest.builder()
        .username(userName)
        .userPoolId(poolId)
        .build();

    AdminGetUserResponse response =
identityProviderClient.adminGetUser(userRequest);
    System.out.println("User status " + response.userStatusAsString());

} catch (CognitoIdentityProviderException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

- Per i dettagli sull'API, consulta la [AdminGetUser](#) sezione AWS SDK for Java 2.x API Reference.

## AdminInitiateAuth

Il seguente esempio di codice mostra come utilizzare `AdminInitiateAuth`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public static AdminInitiateAuthResponse
initiateAuth(CognitoIdentityProviderClient identityProviderClient,
    String clientId, String userName, String password, String userPoolId) {
    try {
        Map<String, String> authParameters = new HashMap<>();
        authParameters.put("USERNAME", userName);
        authParameters.put("PASSWORD", password);

        AdminInitiateAuthRequest authRequest =
AdminInitiateAuthRequest.builder()
            .clientId(clientId)
```

```
        .userPoolId(userPoolId)
        .authParameters(authParameters)
        .authFlow(AuthFlowType.ADMIN_USER_PASSWORD_AUTH)
        .build();

        AdminInitiateAuthResponse response =
identityProviderClient.adminInitiateAuth(authRequest);
        System.out.println("Result Challenge is : " + response.challengeName());
        return response;

    } catch (CognitoIdentityProviderException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }

    return null;
}
```

- Per i dettagli sull'API, consulta la [AdminInitiateAuth](#) sezione AWS SDK for Java 2.x API Reference.

## AdminRespondToAuthChallenge

Il seguente esempio di codice mostra come utilizzare `AdminRespondToAuthChallenge`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
// Respond to an authentication challenge.
public static void adminRespondToAuthChallenge(CognitoIdentityProviderClient
identityProviderClient,
        String userName, String clientId, String mfaCode, String session) {
    System.out.println("SOFTWARE_TOKEN_MFA challenge is generated");
    Map<String, String> challengeResponses = new HashMap<>();
```

```
challengeResponses.put("USERNAME", userName);
challengeResponses.put("SOFTWARE_TOKEN_MFA_CODE", mfaCode);

AdminRespondToAuthChallengeRequest respondToAuthChallengeRequest =
AdminRespondToAuthChallengeRequest.builder()
    .challengeName(ChallengeNameType.SOFTWARE_TOKEN_MFA)
    .clientId(clientId)
    .challengeResponses(challengeResponses)
    .session(session)
    .build();

AdminRespondToAuthChallengeResponse respondToAuthChallengeResult =
identityProviderClient
    .adminRespondToAuthChallenge(respondToAuthChallengeRequest);
System.out.println("respondToAuthChallengeResult.getAuthenticationResult()"
    + respondToAuthChallengeResult.authenticationResult());
}
```

- Per i dettagli sull'API, consulta la [AdminRespondToAuthChallenge](#) sezione AWS SDK for Java 2.x API Reference.

## AssociateSoftwareToken

Il seguente esempio di codice mostra come utilizzare `AssociateSoftwareToken`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public static String getSecretForAppMFA(CognitoIdentityProviderClient
identityProviderClient, String session) {
    AssociateSoftwareTokenRequest softwareTokenRequest =
AssociateSoftwareTokenRequest.builder()
    .session(session)
    .build();
```

```
AssociateSoftwareTokenResponse tokenResponse = identityProviderClient
    .associateSoftwareToken(softwareTokenRequest);
String secretCode = tokenResponse.secretCode();
System.out.println("Enter this token into Google Authenticator");
System.out.println(secretCode);
return tokenResponse.session();
}
```

- Per i dettagli sull'API, consulta la [AssociateSoftwareToken](#) sezione AWS SDK for Java 2.x API Reference.

## ConfirmSignUp

Il seguente esempio di codice mostra come utilizzare `ConfirmSignUp`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public static void confirmSignUp(CognitoIdentityProviderClient
identityProviderClient, String clientId, String code,
String userName) {
    try {
        ConfirmSignUpRequest signUpRequest = ConfirmSignUpRequest.builder()
            .clientId(clientId)
            .confirmationCode(code)
            .username(userName)
            .build();

        identityProviderClient.confirmSignUp(signUpRequest);
        System.out.println(userName + " was confirmed");

    } catch (CognitoIdentityProviderException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

```
}
```

- Per i dettagli sull'API, consulta la [ConfirmSignUp](#) sezione AWS SDK for Java 2.x API Reference.

## CreateUserPool

Il seguente esempio di codice mostra come utilizzare `CreateUserPool`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;
import
    software.amazon.awssdk.services.cognitoidentityprovider.CognitoIdentityProviderClient;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderException;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.CreateUserPoolRequest;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.CreateUserPoolResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateUserPool {
    public static void main(String[] args) {

        final String usage = ""

                Usage:
                <userPoolName>\s
```

```

        Where:
            userPoolName - The name to give your user pool when it's
created.
        """;

    if (args.length != 1) {
        System.out.println(usage);
        System.exit(1);
    }

    String userPoolName = args[0];
    CognitoIdentityProviderClient cognitoClient =
CognitoIdentityProviderClient.builder()
        .region(Region.US_EAST_1)
        .build();

    String id = createPool(cognitoClient, userPoolName);
    System.out.println("User pool ID: " + id);
    cognitoClient.close();
}

public static String createPool(CognitoIdentityProviderClient cognitoClient,
String userPoolName) {
    try {
        CreateUserPoolRequest request = CreateUserPoolRequest.builder()
            .poolName(userPoolName)
            .build();

        CreateUserPoolResponse response = cognitoClient.createUserPool(request);
        return response.userPool().id();

    } catch (CognitoIdentityProviderException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
}
}

```

- Per i dettagli sull'API, consulta la [CreateUserPool](#) sezione AWS SDK for Java 2.x API Reference.



## CreateUserPoolClient

Il seguente esempio di codice mostra come utilizzare `CreateUserPoolClient`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;
import
    software.amazon.awssdk.services.cognitoidentityprovider.CognitoIdentityProviderClient;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderException;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.CreateUserPoolClientRequest;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.CreateUserPoolClientResponse;

/**
 * A user pool client app is an application that authenticates with Amazon
 * Cognito user pools.
 * When you create a user pool, you can configure app clients that allow mobile
 * or web applications
 * to call API operations to authenticate users, manage user attributes and
 * profiles,
 * and implement sign-up and sign-in flows.
 *
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateUserPoolClient {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <clientName> <userPoolId>\s
```

```
        Where:
            clientName - The name for the user pool client to create.
            userPoolId - The ID for the user pool.
        """;

    if (args.length != 2) {
        System.out.println(usage);
        System.exit(1);
    }

    String clientName = args[0];
    String userPoolId = args[1];
    CognitoIdentityProviderClient cognitoClient =
CognitoIdentityProviderClient.builder()
        .region(Region.US_EAST_1)
        .build();

    createPoolClient(cognitoClient, clientName, userPoolId);
    cognitoClient.close();
}

public static void createPoolClient(CognitoIdentityProviderClient cognitoClient,
String clientName,
    String userPoolId) {
    try {
        CreateUserPoolClientRequest request =
CreateUserPoolClientRequest.builder()
            .clientName(clientName)
            .userPoolId(userPoolId)
            .build();

        CreateUserPoolClientResponse response =
cognitoClient.createUserPoolClient(request);
        System.out.println("User pool " + response.userPoolClient().clientName()
+ " created. ID: "
            + response.userPoolClient().clientId());

    } catch (CognitoIdentityProviderException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Per i dettagli sull'API, [CreateUserPoolClient](#) consulta AWS SDK for Java 2.x API Reference.

## ListUserPools

Il seguente esempio di codice mostra come utilizzare `ListUserPools`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;
import
    software.amazon.awssdk.services.cognitoidentityprovider.CognitoIdentityProviderClient;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderException;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.ListUserPoolsResponse;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.ListUserPoolsRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListUserPools {
    public static void main(String[] args) {
        CognitoIdentityProviderClient cognitoClient =
        CognitoIdentityProviderClient.builder()
            .region(Region.US_EAST_1)
            .build();

        listAllUserPools(cognitoClient);
    }
}
```

```
        cognitoClient.close();
    }

    public static void listAllUserPools(CognitoIdentityProviderClient cognitoClient)
    {
        try {
            ListUserPoolsRequest request = ListUserPoolsRequest.builder()
                .maxResults(10)
                .build();

            ListUserPoolsResponse response = cognitoClient.listUserPools(request);
            response.userPools().forEach(userpool -> {
                System.out.println("User pool " + userpool.name() + ", User ID " +
                    userpool.id());
            });

        } catch (CognitoIdentityProviderException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Per i dettagli sull'API, [ListUserPools](#) consulta AWS SDK for Java 2.x API Reference.

## ListUsers

Il seguente esempio di codice mostra come utilizzare `ListUsers`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;
import
software.amazon.awssdk.services.cognitoidentityprovider.CognitoIdentityProviderClient;
```

```
import
software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderException;
import
software.amazon.awssdk.services.cognitoidentityprovider.model.ListUsersRequest;
import
software.amazon.awssdk.services.cognitoidentityprovider.model.ListUsersResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListUsers {
    public static void main(String[] args) {

        final String usage = ""

            Usage:
                <userPoolId>\s

            Where:
                userPoolId - The ID given to your user pool when it's created.
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String userPoolId = args[0];
        CognitoIdentityProviderClient cognitoClient =
CognitoIdentityProviderClient.builder()
            .region(Region.US_EAST_1)
            .build();

        listAllUsers(cognitoClient, userPoolId);
        listUsersFilter(cognitoClient, userPoolId);
        cognitoClient.close();
    }
}
```

```
public static void listAllUsers(CognitoIdentityProviderClient cognitoClient,
String userPoolId) {
    try {
        ListUsersRequest usersRequest = ListUsersRequest.builder()
            .userPoolId(userPoolId)
            .build();

        ListUsersResponse response = cognitoClient.listUsers(usersRequest);
        response.users().forEach(user -> {
            System.out.println("User " + user.username() + " Status " +
user.userStatus() + " Created "
                + user.userCreateDate());
        });

    } catch (CognitoIdentityProviderException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

// Shows how to list users by using a filter.
public static void listUsersFilter(CognitoIdentityProviderClient cognitoClient,
String userPoolId) {

    try {
        String filter = "email = \"tblue@noserver.com\"";
        ListUsersRequest usersRequest = ListUsersRequest.builder()
            .userPoolId(userPoolId)
            .filter(filter)
            .build();

        ListUsersResponse response = cognitoClient.listUsers(usersRequest);
        response.users().forEach(user -> {
            System.out.println("User with filter applied " + user.username() + "
Status " + user.userStatus()
                + " Created " + user.userCreateDate());
        });

    } catch (CognitoIdentityProviderException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Per i dettagli sull'API, [ListUsers](#) consulta AWS SDK for Java 2.x API Reference.

## ResendConfirmationCode

Il seguente esempio di codice mostra come utilizzare `ResendConfirmationCode`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public static void resendConfirmationCode(CognitoIdentityProviderClient
identityProviderClient, String clientId,
    String userName) {
    try {
        ResendConfirmationCodeRequest codeRequest =
ResendConfirmationCodeRequest.builder()
            .clientId(clientId)
            .username(userName)
            .build();

        ResendConfirmationCodeResponse response =
identityProviderClient.resendConfirmationCode(codeRequest);
        System.out.println("Method of delivery is " +
response.codeDeliveryDetails().deliveryMediumAsString());

    } catch (CognitoIdentityProviderException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Per i dettagli sull'API, [ResendConfirmationCode](#) consulta AWS SDK for Java 2.x API Reference.

## SignUp

Il seguente esempio di codice mostra come utilizzare SignUp.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public static void signUp(CognitoIdentityProviderClient identityProviderClient,
String clientId, String userName,
    String password, String email) {
    AttributeType userAttrs = AttributeType.builder()
        .name("email")
        .value(email)
        .build();

    List<AttributeType> userAttrsList = new ArrayList<>();
    userAttrsList.add(userAttrs);
    try {
        SignUpRequest signUpRequest = SignUpRequest.builder()
            .userAttributes(userAttrsList)
            .username(userName)
            .clientId(clientId)
            .password(password)
            .build();

        identityProviderClient.signUp(signUpRequest);
        System.out.println("User has been signed up ");

    } catch (CognitoIdentityProviderException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Per i dettagli sull'API, [SignUp](#) consulta AWS SDK for Java 2.x API Reference.



## VerifySoftwareToken

Il seguente esempio di codice mostra come utilizzare `VerifySoftwareToken`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
// Verify the TOTP and register for MFA.
public static void verifyTOTP(CognitoIdentityProviderClient
identityProviderClient, String session, String code) {
    try {
        VerifySoftwareTokenRequest tokenRequest =
VerifySoftwareTokenRequest.builder()
            .userCode(code)
            .session(session)
            .build();

        VerifySoftwareTokenResponse verifyResponse =
identityProviderClient.verifySoftwareToken(tokenRequest);
        System.out.println("The status of the token is " +
verifyResponse.statusAsString());

    } catch (CognitoIdentityProviderException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Per i dettagli sull'API, [VerifySoftwareToken](#) consulta AWS SDK for Java 2.x API Reference.

## Scenari

Registrazione di un utente a un pool di utenti che richiede l'autenticazione MFA

L'esempio di codice seguente mostra come:

- Registra e conferma un utente con nome utente, password e indirizzo e-mail.
- Configura l'autenticazione a più fattori associando un'applicazione MFA all'utente.
- Accedi utilizzando una password e un codice MFA.

## SDK per Java 2.x

### Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;
import
    software.amazon.awssdk.services.cognitoidentityprovider.CognitoIdentityProviderClient;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.AdminGetUserRequest;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.AdminGetUserResponse;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.AdminInitiateAuthRequest;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.AdminInitiateAuthResponse;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.AdminRespondToAuthChallengeRequest;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.AdminRespondToAuthChallengeResponse;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.AssociateSoftwareTokenRequest;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.AssociateSoftwareTokenResponse;
import software.amazon.awssdk.services.cognitoidentityprovider.model.AttributeType;
import software.amazon.awssdk.services.cognitoidentityprovider.model.AuthFlowType;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.ChallengeNameType;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderException;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.ConfirmSignUpRequest;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.ResendConfirmationCodeRequest;
```

```
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.ResendConfirmationCodeResponse;
import software.amazon.awssdk.services.cognitoidentityprovider.model.SignUpRequest;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.VerifySoftwareTokenRequest;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.VerifySoftwareTokenResponse;
import java.security.InvalidKeyException;
import java.security.NoSuchAlgorithmException;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.Scanner;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * TIP: To set up the required user pool, run the AWS Cloud Development Kit (AWS
 * CDK) script provided in this GitHub repo at
 * resources/cdk/cognito\_scenario\_user\_pool\_with\_mfa.
 *
 * This code example performs the following operations:
 *
 * 1. Invokes the signUp method to sign up a user.
 * 2. Invokes the adminGetUser method to get the user's confirmation status.
 * 3. Invokes the ResendConfirmationCode method if the user requested another
 * code.
 * 4. Invokes the confirmSignUp method.
 * 5. Invokes the AdminInitiateAuth to sign in. This results in being prompted
 * to set up TOTP (time-based one-time password). (The response is
 * "ChallengeName": "MFA_SETUP").
 * 6. Invokes the AssociateSoftwareToken method to generate a TOTP MFA private
 * key. This can be used with Google Authenticator.
 * 7. Invokes the VerifySoftwareToken method to verify the TOTP and register for
 * MFA.
 * 8. Invokes the AdminInitiateAuth to sign in again. This results in being
 * prompted to submit a TOTP (Response: "ChallengeName": "SOFTWARE_TOKEN_MFA").
 * 9. Invokes the AdminRespondToAuthChallenge to get back a token.
```

```
*/

public class CognitoMVP {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    public static void main(String[] args) throws NoSuchAlgorithmException,
    InvalidKeyException {
        final String usage = ""

            Usage:
                <clientId> <poolId>

            Where:
                clientId - The app client Id value that you can get from the AWS
CDK script.
                poolId - The pool Id that you can get from the AWS CDK script.\s
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String clientId = args[0];
        String poolId = args[1];
        CognitoIdentityProviderClient identityProviderClient =
CognitoIdentityProviderClient.builder()
            .region(Region.US_EAST_1)
            .build();

        System.out.println(DASHES);
        System.out.println("Welcome to the Amazon Cognito example scenario.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("**** Enter your user name");
        Scanner in = new Scanner(System.in);
        String userName = in.nextLine();

        System.out.println("**** Enter your password");
        String password = in.nextLine();

        System.out.println("**** Enter your email");
        String email = in.nextLine();
    }
}
```

```
System.out.println("1. Signing up " + userName);
signUp(identityProviderClient, clientId, userName, password, email);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("2. Getting " + userName + " in the user pool");
getAdminUser(identityProviderClient, userName, poolId);

System.out
    .println("*** Conformation code sent to " + userName + ". Would you
like to send a new code? (Yes/No)");
System.out.println(DASHES);

System.out.println(DASHES);
String ans = in.nextLine();

if (ans.compareTo("Yes") == 0) {
    resendConfirmationCode(identityProviderClient, clientId, userName);
    System.out.println("3. Sending a new confirmation code");
}
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("4. Enter confirmation code that was emailed");
String code = in.nextLine();
confirmSignUp(identityProviderClient, clientId, code, userName);
System.out.println("Rechecking the status of " + userName + " in the user
pool");
getAdminUser(identityProviderClient, userName, poolId);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("5. Invokes the initiateAuth to sign in");
AdminInitiateAuthResponse authResponse =
initiateAuth(identityProviderClient, clientId, userName, password,
    poolId);
String mySession = authResponse.session();
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("6. Invokes the AssociateSoftwareToken method to generate
a TOTP key");
String newSession = getSecretForAppMFA(identityProviderClient, mySession);
```

```
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("*** Enter the 6-digit code displayed in Google
Authenticator");
        String myCode = in.nextLine();
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Verify the TOTP and register for MFA");
        verifyTOTP(identityProviderClient, newSession, myCode);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("8. Re-enter a 6-digit code displayed in Google
Authenticator");
        String mfaCode = in.nextLine();
        AdminInitiateAuthResponse authResponse1 =
initiateAuth(identityProviderClient, clientId, userName, password,
                poolId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("9. Invokes the AdminRespondToAuthChallenge");
        String session2 = authResponse1.session();
        adminRespondToAuthChallenge(identityProviderClient, userName, clientId,
mfaCode, session2);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("All Amazon Cognito operations were successfully
performed");
        System.out.println(DASHES);
    }

    // Respond to an authentication challenge.
    public static void adminRespondToAuthChallenge(CognitoIdentityProviderClient
identityProviderClient,
            String userName, String clientId, String mfaCode, String session) {
        System.out.println("SOFTWARE_TOKEN_MFA challenge is generated");
        Map<String, String> challengeResponses = new HashMap<>();

        challengeResponses.put("USERNAME", userName);
        challengeResponses.put("SOFTWARE_TOKEN_MFA_CODE", mfaCode);
    }
}
```

```

        AdminRespondToAuthChallengeRequest respondToAuthChallengeRequest =
AdminRespondToAuthChallengeRequest.builder()
    .challengeName(ChallengeNameType.SOFTWARE_TOKEN_MFA)
    .clientId(clientId)
    .challengeResponses(challengeResponses)
    .session(session)
    .build();

        AdminRespondToAuthChallengeResponse respondToAuthChallengeResult =
identityProviderClient
    .adminRespondToAuthChallenge(respondToAuthChallengeRequest);
        System.out.println("respondToAuthChallengeResult.getAuthenticationResult()"
    + respondToAuthChallengeResult.authenticationResult());
    }

    // Verify the TOTP and register for MFA.
    public static void verifyTOTP(CognitoIdentityProviderClient
identityProviderClient, String session, String code) {
        try {
            VerifySoftwareTokenRequest tokenRequest =
VerifySoftwareTokenRequest.builder()
                .userCode(code)
                .session(session)
                .build();

            VerifySoftwareTokenResponse verifyResponse =
identityProviderClient.verifySoftwareToken(tokenRequest);
            System.out.println("The status of the token is " +
verifyResponse.statusAsString());

        } catch (CognitoIdentityProviderException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static AdminInitiateAuthResponse
initiateAuth(CognitoIdentityProviderClient identityProviderClient,
        String clientId, String userName, String password, String userPoolId) {
        try {
            Map<String, String> authParameters = new HashMap<>();
            authParameters.put("USERNAME", userName);
            authParameters.put("PASSWORD", password);

```

```
        AdminInitiateAuthRequest authRequest =
AdminInitiateAuthRequest.builder()
        .clientId(clientId)
        .userPoolId(userPoolId)
        .authParameters(authParameters)
        .authFlow(AuthFlowType.ADMIN_USER_PASSWORD_AUTH)
        .build();

        AdminInitiateAuthResponse response =
identityProviderClient.adminInitiateAuth(authRequest);
        System.out.println("Result Challenge is : " + response.challengeName());
        return response;

    } catch (CognitoIdentityProviderException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }

    return null;
}

public static String getSecretForAppMFA(CognitoIdentityProviderClient
identityProviderClient, String session) {
    AssociateSoftwareTokenRequest softwareTokenRequest =
AssociateSoftwareTokenRequest.builder()
        .session(session)
        .build();

    AssociateSoftwareTokenResponse tokenResponse = identityProviderClient
        .associateSoftwareToken(softwareTokenRequest);
    String secretCode = tokenResponse.secretCode();
    System.out.println("Enter this token into Google Authenticator");
    System.out.println(secretCode);
    return tokenResponse.session();
}

public static void confirmSignUp(CognitoIdentityProviderClient
identityProviderClient, String clientId, String code,
String userName) {
    try {
        ConfirmSignUpRequest signUpRequest = ConfirmSignUpRequest.builder()
            .clientId(clientId)
            .confirmationCode(code)
```



```
        .username(userName)
        .build();

    identityProviderClient.confirmSignUp(signUpRequest);
    System.out.println(userName + " was confirmed");

} catch (CognitoIdentityProviderException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}

public static void resendConfirmationCode(CognitoIdentityProviderClient
identityProviderClient, String clientId,
    String userName) {
    try {
        ResendConfirmationCodeRequest codeRequest =
ResendConfirmationCodeRequest.builder()
            .clientId(clientId)
            .username(userName)
            .build();

        ResendConfirmationCodeResponse response =
identityProviderClient.resendConfirmationCode(codeRequest);
        System.out.println("Method of delivery is " +
response.codeDeliveryDetails().deliveryMediumAsString());

    } catch (CognitoIdentityProviderException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void signUp(CognitoIdentityProviderClient identityProviderClient,
String clientId, String userName,
    String password, String email) {
    AttributeType userAttrs = AttributeType.builder()
        .name("email")
        .value(email)
        .build();

    List<AttributeType> userAttrsList = new ArrayList<>();
    userAttrsList.add(userAttrs);
    try {
```

```

        SignUpRequest signUpRequest = SignUpRequest.builder()
            .userAttributes(userAttrsList)
            .username(userName)
            .clientId(clientId)
            .password(password)
            .build();

        identityProviderClient.signUp(signUpRequest);
        System.out.println("User has been signed up ");

    } catch (CognitoIdentityProviderException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void getAdminUser(CognitoIdentityProviderClient
identityProviderClient, String userName,
    String poolId) {
    try {
        AdminGetUserRequest userRequest = AdminGetUserRequest.builder()
            .username(userName)
            .userPoolId(poolId)
            .build();

        AdminGetUserResponse response =
identityProviderClient.adminGetUser(userRequest);
        System.out.println("User status " + response.userStatusAsString());

    } catch (CognitoIdentityProviderException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
}

```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella Documentazione di riferimento delle API AWS SDK for Java 2.x .
  - [AdminGetUser](#)
  - [AdminInitiateAuth](#)
  - [AdminRespondToAuthChallenge](#)

- [AssociateSoftwareToken](#)
- [ConfirmDevice](#)
- [ConfirmSignUp](#)
- [InitiateAuth](#)
- [ListUsers](#)
- [ResendConfirmationCode](#)
- [RespondToAuthChallenge](#)
- [SignUp](#)
- [VerifySoftwareToken](#)

## Esempi di Amazon Comprehend con SDK for Java 2.x

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando Amazon Comprehend. AWS SDK for Java 2.x

Le operazioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le operazioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati e negli esempi tra servizi.

Scenari: esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio.

Ogni esempio include un collegamento a GitHub, dove puoi trovare istruzioni su come configurare ed eseguire il codice nel contesto.

### Argomenti

- [Azioni](#)

### Azioni

#### **CreateDocumentClassifier**

Il seguente esempio di codice mostra come utilizzare `CreateDocumentClassifier`.

## SDK per Java 2.x

 Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.comprehend.ComprehendClient;
import software.amazon.awssdk.services.comprehend.model.ComprehendException;
import
    software.amazon.awssdk.services.comprehend.model.CreateDocumentClassifierRequest;
import
    software.amazon.awssdk.services.comprehend.model.CreateDocumentClassifierResponse;
import
    software.amazon.awssdk.services.comprehend.model.DocumentClassifierInputDataConfig;

/**
 * Before running this code example, you can setup the necessary resources, such
 * as the CSV file and IAM Roles, by following this document:
 * https://aws.amazon.com/blogs/machine-learning/building-a-custom-classifier-using-
amazon-comprehend/
 *
 * Also, set up your development environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DocumentClassifierDemo {
    public static void main(String[] args) {
        final String usage = ""

            Usage:    <dataAccessRoleArn> <s3Uri> <documentClassifierName>

            Where:
                dataAccessRoleArn - The ARN value of the role used for this
operation.

                s3Uri - The Amazon S3 bucket that contains the CSV file.
                documentClassifierName - The name of the document classifier.

            """;
```

```
    if (args.length != 3) {
        System.out.println(usage);
        System.exit(1);
    }

    String dataAccessRoleArn = args[0];
    String s3Uri = args[1];
    String documentClassifierName = args[2];

    Region region = Region.US_EAST_1;
    ComprehendClient comClient = ComprehendClient.builder()
        .region(region)
        .build();

    createDocumentClassifier(comClient, dataAccessRoleArn, s3Uri,
documentClassifierName);
    comClient.close();
}

public static void createDocumentClassifier(ComprehendClient comClient, String
dataAccessRoleArn, String s3Uri,
    String documentClassifierName) {
    try {
        DocumentClassifierInputDataConfig config =
DocumentClassifierInputDataConfig.builder()
            .s3Uri(s3Uri)
            .build();

        CreateDocumentClassifierRequest createDocumentClassifierRequest =
CreateDocumentClassifierRequest.builder()
            .documentClassifierName(documentClassifierName)
            .dataAccessRoleArn(dataAccessRoleArn)
            .languageCode("en")
            .inputDataConfig(config)
            .build();

        CreateDocumentClassifierResponse createDocumentClassifierResult =
comClient
            .createDocumentClassifier(createDocumentClassifierRequest);
        String documentClassifierArn =
createDocumentClassifierResult.documentClassifierArn();
        System.out.println("Document Classifier ARN: " + documentClassifierArn);
    }
}
```

```
        } catch (ComprehendException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Per i dettagli sull'API, [CreateDocumentClassifier](#) consulta AWS SDK for Java 2.x API Reference.

## DetectDominantLanguage

Il seguente esempio di codice mostra come utilizzare `DetectDominantLanguage`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.comprehend.ComprehendClient;
import software.amazon.awssdk.services.comprehend.model.ComprehendException;
import
    software.amazon.awssdk.services.comprehend.model.DetectDominantLanguageRequest;
import
    software.amazon.awssdk.services.comprehend.model.DetectDominantLanguageResponse;
import software.amazon.awssdk.services.comprehend.model.DominantLanguage;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DetectLanguage {
```

```
public static void main(String[] args) {
    // Specify French text - "It is raining today in Seattle".
    String text = "Il pleut aujourd'hui à Seattle";
    Region region = Region.US_EAST_1;

    ComprehendClient comClient = ComprehendClient.builder()
        .region(region)
        .build();

    System.out.println("Calling DetectDominantLanguage");
    detectTheDominantLanguage(comClient, text);
    comClient.close();
}

public static void detectTheDominantLanguage(ComprehendClient comClient, String
text) {
    try {
        DetectDominantLanguageRequest request =
DetectDominantLanguageRequest.builder()
            .text(text)
            .build();

        DetectDominantLanguageResponse resp =
comClient.detectDominantLanguage(request);
        List<DominantLanguage> allLanList = resp.languages();
        for (DominantLanguage lang : allLanList) {
            System.out.println("Language is " + lang.languageCode());
        }

    } catch (ComprehendException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Per i dettagli sull'API, [DetectDominantLanguage](#) consulta AWS SDK for Java 2.x API Reference.

## DetectEntities

Il seguente esempio di codice mostra come utilizzare `DetectEntities`.

## SDK per Java 2.x

 Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.comprehend.ComprehendClient;
import software.amazon.awssdk.services.comprehend.model.DetectEntitiesRequest;
import software.amazon.awssdk.services.comprehend.model.DetectEntitiesResponse;
import software.amazon.awssdk.services.comprehend.model.Entity;
import software.amazon.awssdk.services.comprehend.model.ComprehendException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DetectEntities {
    public static void main(String[] args) {
        String text = "Amazon.com, Inc. is located in Seattle, WA and was founded
        July 5th, 1994 by Jeff Bezos, allowing customers to buy everything from books to
        blenders. Seattle is north of Portland and south of Vancouver, BC. Other notable
        Seattle - based companies are Starbucks and Boeing.";
        Region region = Region.US_EAST_1;
        ComprehendClient comClient = ComprehendClient.builder()
            .region(region)
            .build();

        System.out.println("Calling DetectEntities");
        detectAllEntities(comClient, text);
        comClient.close();
    }

    public static void detectAllEntities(ComprehendClient comClient, String text) {
        try {
```



```
        DetectEntitiesRequest detectEntitiesRequest =
DetectEntitiesRequest.builder()
    .text(text)
    .languageCode("en")
    .build();

        DetectEntitiesResponse detectEntitiesResult =
comClient.detectEntities(detectEntitiesRequest);
        List<Entity> entList = detectEntitiesResult.entities();
        for (Entity entity : entList) {
            System.out.println("Entity text is " + entity.text());
        }

    } catch (ComprehendException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Per i dettagli sull'API, [DetectEntities](#) consulta AWS SDK for Java 2.x API Reference.

## DetectKeyPhrases

Il seguente esempio di codice mostra come utilizzare `DetectKeyPhrases`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.comprehend.ComprehendClient;
import software.amazon.awssdk.services.comprehend.model.DetectKeyPhrasesRequest;
import software.amazon.awssdk.services.comprehend.model.DetectKeyPhrasesResponse;
import software.amazon.awssdk.services.comprehend.model.KeyPhrase;
import software.amazon.awssdk.services.comprehend.model.ComprehendException;
import java.util.List;
```

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DetectKeyPhrases {
    public static void main(String[] args) {
        String text = "Amazon.com, Inc. is located in Seattle, WA and was founded
        July 5th, 1994 by Jeff Bezos, allowing customers to buy everything from books to
        blenders. Seattle is north of Portland and south of Vancouver, BC. Other notable
        Seattle - based companies are Starbucks and Boeing.";
        Region region = Region.US_EAST_1;
        ComprehendClient comClient = ComprehendClient.builder()
            .region(region)
            .build();

        System.out.println("Calling DetectKeyPhrases");
        detectAllKeyPhrases(comClient, text);
        comClient.close();
    }

    public static void detectAllKeyPhrases(ComprehendClient comClient, String text)
    {
        try {
            DetectKeyPhrasesRequest detectKeyPhrasesRequest =
            DetectKeyPhrasesRequest.builder()
                .text(text)
                .languageCode("en")
                .build();

            DetectKeyPhrasesResponse detectKeyPhrasesResult =
            comClient.detectKeyPhrases(detectKeyPhrasesRequest);
            List<KeyPhrase> phraseList = detectKeyPhrasesResult.keyPhrases();
            for (KeyPhrase keyPhrase : phraseList) {
                System.out.println("Key phrase text is " + keyPhrase.text());
            }
        } catch (ComprehendException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

```
    }  
  }  
}
```

- Per i dettagli sull'API, [DetectKeyPhrases](#) consulta AWS SDK for Java 2.x API Reference.

## DetectSentiment

Il seguente esempio di codice mostra come utilizzare `DetectSentiment`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;  
import software.amazon.awssdk.services.comprehend.ComprehendClient;  
import software.amazon.awssdk.services.comprehend.model.ComprehendException;  
import software.amazon.awssdk.services.comprehend.model.DetectSentimentRequest;  
import software.amazon.awssdk.services.comprehend.model.DetectSentimentResponse;  
  
/**  
 * Before running this Java V2 code example, set up your development  
 * environment, including your credentials.  
 *  
 * For more information, see the following documentation topic:  
 *  
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html  
 */  
public class DetectSentiment {  
    public static void main(String[] args) {  
        String text = "Amazon.com, Inc. is located in Seattle, WA and was founded  
        July 5th, 1994 by Jeff Bezos, allowing customers to buy everything from books to  
        blenders. Seattle is north of Portland and south of Vancouver, BC. Other notable  
        Seattle - based companies are Starbucks and Boeing.";  
        Region region = Region.US_EAST_1;  
        ComprehendClient comClient = ComprehendClient.builder()
```

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

        System.out.println("Calling DetectSentiment");
        detectSentiments(comClient, text);
        comClient.close();
    }

    public static void detectSentiments(ComprehendClient comClient, String text) {
        try {
            DetectSentimentRequest detectSentimentRequest =
                DetectSentimentRequest.builder()
                    .text(text)
                    .languageCode("en")
                    .build();

            DetectSentimentResponse detectSentimentResult =
                comClient.detectSentiment(detectSentimentRequest);
            System.out.println("The Neutral value is " +
                detectSentimentResult.sentimentScore().neutral());

        } catch (ComprehendException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Per i dettagli sull'API, [DetectSentiment](#) consulta AWS SDK for Java 2.x API Reference.

## DetectSyntax

Il seguente esempio di codice mostra come utilizzare DetectSyntax.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.comprehend.ComprehendClient;
import software.amazon.awssdk.services.comprehend.model.ComprehendException;
import software.amazon.awssdk.services.comprehend.model.DetectSyntaxRequest;
import software.amazon.awssdk.services.comprehend.model.DetectSyntaxResponse;
import software.amazon.awssdk.services.comprehend.model.SyntaxToken;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DetectSyntax {
    public static void main(String[] args) {
        String text = "Amazon.com, Inc. is located in Seattle, WA and was founded
        July 5th, 1994 by Jeff Bezos, allowing customers to buy everything from books to
        blenders. Seattle is north of Portland and south of Vancouver, BC. Other notable
        Seattle - based companies are Starbucks and Boeing.";
        Region region = Region.US_EAST_1;
        ComprehendClient comClient = ComprehendClient.builder()
            .region(region)
            .build();

        System.out.println("Calling DetectSyntax");
        detectAllSyntax(comClient, text);
        comClient.close();
    }

    public static void detectAllSyntax(ComprehendClient comClient, String text) {
        try {
            DetectSyntaxRequest detectSyntaxRequest = DetectSyntaxRequest.builder()
                .text(text)
                .languageCode("en")
                .build();

            DetectSyntaxResponse detectSyntaxResult =
comClient.detectSyntax(detectSyntaxRequest);
            List<SyntaxToken> syntaxTokens = detectSyntaxResult.syntaxTokens();
            for (SyntaxToken token : syntaxTokens) {
```

```
        System.out.println("Language is " + token.text());
        System.out.println("Part of speech is " +
token.partOfSpeech().tagAsString());
    }

    } catch (ComprehendException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Per i dettagli sull'API, [DetectSyntax](#) consulta AWS SDK for Java 2.x API Reference.

## Esempi di DynamoDB con SDK for Java 2.x

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando AWS SDK for Java 2.x con DynamoDB.

Le operazioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le operazioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati e negli esempi tra servizi.

Scenari: esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio.

Ogni esempio include un collegamento a GitHub, dove è possibile trovare istruzioni su come configurare ed eseguire il codice nel contesto.

Nozioni di base

Hello DynamoDB

Gli esempi di codice seguenti mostrano come iniziare a utilizzare DynamoDB.

## SDK per Java 2.x

 Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.ListTablesRequest;
import software.amazon.awssdk.services.dynamodb.model.ListTablesResponse;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListTables {
    public static void main(String[] args) {
        System.out.println("Listing your Amazon DynamoDB tables:\n");
        Region region = Region.US_EAST_1;
        DynamoDbClient ddb = DynamoDbClient.builder()
            .region(region)
            .build();
        listAllTables(ddb);
        ddb.close();
    }

    public static void listAllTables(DynamoDbClient ddb) {
        boolean moreTables = true;
        String lastName = null;

        while (moreTables) {
            try {
                ListTablesResponse response = null;
                if (lastName == null) {
```

```
        ListTablesRequest request = ListTablesRequest.builder().build();
        response = ddb.listTables(request);
    } else {
        ListTablesRequest request = ListTablesRequest.builder()
            .exclusiveStartTableName(lastName).build();
        response = ddb.listTables(request);
    }

    List<String> tableNames = response.tableNames();
    if (tableNames.size() > 0) {
        for (String curName : tableNames) {
            System.out.format("* %s\n", curName);
        }
    } else {
        System.out.println("No tables found!");
        System.exit(0);
    }

    lastName = response.lastEvaluatedTableName();
    if (lastName == null) {
        moreTables = false;
    }

    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
System.out.println("\nDone!");
}
```

- Per i dettagli sull'API, [ListTables](#) consulta AWS SDK for Java 2.x API Reference.

## Argomenti

- [Azioni](#)
- [Scenari](#)
- [Esempi serverless](#)



## Azioni

### BatchGetItem

Il seguente esempio di codice mostra come utilizzare `BatchGetItem`.

SDK per Java 2.x

#### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

mostra come ottenere articoli in batch utilizzando il client di servizio.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.BatchGetItemRequest;
import software.amazon.awssdk.services.dynamodb.model.BatchGetItemResponse;
import software.amazon.awssdk.services.dynamodb.model.KeysAndAttributes;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

/**
 * Before running this Java V2 code example, set up your development environment,
 * including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class BatchReadItems {
    public static void main(String[] args){
        final String usage = ""

                Usage:
                <tableName>

                Where:
                tableName - The Amazon DynamoDB table (for example, Music).\s
```

```

        """;

        String tableName = "Music";
        Region region = Region.US_EAST_1;
        DynamoDbClient dynamoDbClient = DynamoDbClient.builder()
            .region(region)
            .build();

        getBatchItems(dynamoDbClient, tableName);
    }

    public static void getBatchItems(DynamoDbClient dynamoDbClient, String
tableName) {
        // Define the primary key values for the items you want to retrieve.
        Map<String, AttributeValue> key1 = new HashMap<>();
        key1.put("Artist", AttributeValue.builder().s("Artist1").build());

        Map<String, AttributeValue> key2 = new HashMap<>();
        key2.put("Artist", AttributeValue.builder().s("Artist2").build());

        // Construct the batchGetItem request.
        Map<String, KeysAndAttributes> requestItems = new HashMap<>();
        requestItems.put(tableName, KeysAndAttributes.builder()
            .keys(List.of(key1, key2))
            .projectionExpression("Artist, SongTitle")
            .build());

        BatchGetItemRequest batchGetItemRequest = BatchGetItemRequest.builder()
            .requestItems(requestItems)
            .build();

        // Make the batchGetItem request.
        BatchGetItemResponse batchGetItemResponse =
dynamoDbClient.batchGetItem(batchGetItemRequest);

        // Extract and print the retrieved items.
        Map<String, List<Map<String, AttributeValue>>> responses =
batchGetItemResponse.responses();
        if (responses.containsKey(tableName)) {
            List<Map<String, AttributeValue>> musicItems = responses.get(tableName);
            for (Map<String, AttributeValue> item : musicItems) {
                System.out.println("Artist: " + item.get("Artist").s() +
                    ", SongTitle: " + item.get("SongTitle").s());
            }
        }
    }

```

```
        } else {
            System.out.println("No items retrieved.");
        }
    }
}
```

mostra come ottenere articoli in batch utilizzando il client di servizio e un impaginatore.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.BatchGetItemRequest;
import software.amazon.awssdk.services.dynamodb.model.KeysAndAttributes;
import java.util.Collections;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

public class BatchGetItemsPaginator {

    public static void main(String[] args){
        final String usage = ""

            Usage:
                <tableName>

            Where:
                tableName - The Amazon DynamoDB table (for example, Music).\s
            """;

        String tableName = "Music";
        Region region = Region.US_EAST_1;
        DynamoDbClient dynamoDbClient = DynamoDbClient.builder()
            .region(region)
            .build();

        getBatchItemsPaginator(dynamoDbClient, tableName) ;
    }

    public static void getBatchItemsPaginator(DynamoDbClient dynamoDbClient, String
tableName) {
        // Define the primary key values for the items you want to retrieve.
```

```

Map<String, AttributeValue> key1 = new HashMap<>();
key1.put("Artist", AttributeValue.builder().s("Artist1").build());

Map<String, AttributeValue> key2 = new HashMap<>();
key2.put("Artist", AttributeValue.builder().s("Artist2").build());

// Construct the batchGetItem request.
Map<String, KeysAndAttributes> requestItems = new HashMap<>();
requestItems.put(tableName, KeysAndAttributes.builder()
    .keys(List.of(key1, key2))
    .projectionExpression("Artist, SongTitle")
    .build());

BatchGetItemRequest batchGetItemRequest = BatchGetItemRequest.builder()
    .requestItems(requestItems)
    .build();

// Use batchGetItemPaginator for paginated requests.
dynamoDbClient.batchGetItemPaginator(batchGetItemRequest).stream()
    .flatMap(response -> response.responses().getOrDefault(tableName,
Collections.emptyList()).stream())
    .forEach(item -> {
        System.out.println("Artist: " + item.get("Artist").s() +
            ", SongTitle: " + item.get("SongTitle").s());
    });
}
}

```

- Per i dettagli sull'API, consulta la sezione AWS SDK for Java 2.x API [BatchGetItem](#) Reference.

## BatchWriteItem

Il seguente esempio di codice mostra come utilizzare `BatchWriteItem`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Inserisce molti elementi in una tabella utilizzando il client di servizio.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.BatchWriteItemRequest;
import software.amazon.awssdk.services.dynamodb.model.BatchWriteItemResponse;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.PutRequest;
import software.amazon.awssdk.services.dynamodb.model.WriteRequest;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

/**
 * Before running this Java V2 code example, set up your development environment,
 * including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class BatchWriteItems {
    public static void main(String[] args){
        final String usage = ""

                Usage:
                <tableName>

                Where:
                tableName - The Amazon DynamoDB table (for example, Music).\s
                """;

        String tableName = "Music";
        Region region = Region.US_EAST_1;
        DynamoDbClient dynamoDbClient = DynamoDbClient.builder()
                .region(region)
                .build();

        addBatchItems(dynamoDbClient, tableName);
    }
}
```

```
public static void addBatchItems(DynamoDbClient dynamoDbClient, String
tableName) {
    // Specify the updates you want to perform.
    List<WriteRequest> writeRequests = new ArrayList<>();

    // Set item 1.
    Map<String, AttributeValue> item1Attributes = new HashMap<>();
    item1Attributes.put("Artist",
AttributeValue.builder().s("Artist1").build());
    item1Attributes.put("Rating", AttributeValue.builder().s("5").build());
    item1Attributes.put("Comments", AttributeValue.builder().s("Great
song!").build());
    item1Attributes.put("SongTitle",
AttributeValue.builder().s("SongTitle1").build());

writeRequests.add(WriteRequest.builder().putRequest(PutRequest.builder().item(item1Attribut

    // Set item 2.
    Map<String, AttributeValue> item2Attributes = new HashMap<>();
    item2Attributes.put("Artist",
AttributeValue.builder().s("Artist2").build());
    item2Attributes.put("Rating", AttributeValue.builder().s("4").build());
    item2Attributes.put("Comments", AttributeValue.builder().s("Nice
melody.").build());
    item2Attributes.put("SongTitle",
AttributeValue.builder().s("SongTitle2").build());

writeRequests.add(WriteRequest.builder().putRequest(PutRequest.builder().item(item2Attribut

    try {
        // Create the BatchWriteItemRequest.
        BatchWriteItemRequest batchWriteItemRequest =
BatchWriteItemRequest.builder()
            .requestItems(Map.of(tableName, writeRequests))
            .build();

        // Execute the BatchWriteItem operation.
        BatchWriteItemResponse batchWriteItemResponse =
dynamoDbClient.batchWriteItem(batchWriteItemRequest);

        // Process the response.
        System.out.println("Batch write successful: " + batchWriteItemResponse);

    } catch (DynamoDbException e) {
```

```
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
}
```

Inserisce molti elementi in una tabella utilizzando il client avanzato.

```
import com.example.dynamodb.Customer;
import com.example.dynamodb.Music;
import software.amazon.awssdk.enhanced.dynamodb.DynamoDbEnhancedClient;
import software.amazon.awssdk.enhanced.dynamodb.DynamoDbTable;
import software.amazon.awssdk.enhanced.dynamodb.Key;
import software.amazon.awssdk.enhanced.dynamodb.TableSchema;
import software.amazon.awssdk.enhanced.dynamodb.model.BatchWriteItemEnhancedRequest;
import software.amazon.awssdk.enhanced.dynamodb.model.WriteBatch;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import java.time.Instant;
import java.time.LocalDate;
import java.time.LocalDateTime;
import java.time.ZoneOffset;

/*
 * Before running this code example, create an Amazon DynamoDB table named Customer
 * with these columns:
 * - id - the id of the record that is the key
 * - custName - the customer name
 * - email - the email value
 * - registrationDate - an instant value when the item was added to the table
 *
 * Also, ensure that you have set up your development environment, including your
 * credentials.
 *
 * For information, see this documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class EnhancedBatchWriteItems {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
```

```
DynamoDbClient ddb = DynamoDbClient.builder()
    .region(region)
    .build();
DynamoDbEnhancedClient enhancedClient =
DynamoDbEnhancedClient.builder()
    .dynamoDbClient(ddb)
    .build();
putBatchRecords(enhancedClient);
ddb.close();
}

public static void putBatchRecords(DynamoDbEnhancedClient enhancedClient) {
    try {
        DynamoDbTable<Customer> customerMappedTable =
enhancedClient.table("Customer",
                        TableSchema.fromBean(Customer.class));
        DynamoDbTable<Music> musicMappedTable =
enhancedClient.table("Music",
                        TableSchema.fromBean(Music.class));
        LocalDate localDate = LocalDate.parse("2020-04-07");
        LocalDateTime localDateTime = localDate.atStartOfDay();
        Instant instant = localDateTime.toInstant(ZoneOffset.UTC);

        Customer record2 = new Customer();
        record2.setCustName("Fred Pink");
        record2.setId("id110");
        record2.setEmail("fredp@noserver.com");
        record2.setRegistrationDate(instant);

        Customer record3 = new Customer();
        record3.setCustName("Susan Pink");
        record3.setId("id120");
        record3.setEmail("spink@noserver.com");
        record3.setRegistrationDate(instant);

        Customer record4 = new Customer();
        record4.setCustName("Jerry orange");
        record4.setId("id101");
        record4.setEmail("jorange@noserver.com");
        record4.setRegistrationDate(instant);

        BatchWriteItemEnhancedRequest batchWriteItemEnhancedRequest
= BatchWriteItemEnhancedRequest
                                .builder()
```



```

        .writeBatches(

WriteBatch.builder(Customer.class) // add items to the Customer

    // table

    .mappedTableResource(customerMappedTable)

    .addPutItem(builder -> builder.item(record2))

    .addPutItem(builder -> builder.item(record3))

    .addPutItem(builder -> builder.item(record4))

                                                                    .build(),

WriteBatch.builder(Music.class) // delete an item from the Music

    // table

    .mappedTableResource(musicMappedTable)

    .addDeleteItem(builder -> builder.key(

        Key.builder().partitionValue(

            "Famous Band")

            .build()))

                                                                    .build())

                                                                    .build();

                                                                    // Add three items to the Customer table and delete one item
from the Music
                                                                    // table.

enhancedClient.batchWriteItem(batchWriteItemEnhancedRequest);
    System.out.println("done");

    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
}
}

```

- Per i dettagli sull'API, vedere [BatchWriteItem](#) in AWS SDK for Java 2.x API Reference.

## CreateTable

Il seguente esempio di codice mostra come utilizzare `CreateTable`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeDefinition;
import software.amazon.awssdk.services.dynamodb.model.CreateTableRequest;
import software.amazon.awssdk.services.dynamodb.model.CreateTableResponse;
import software.amazon.awssdk.services.dynamodb.model.DescribeTableRequest;
import software.amazon.awssdk.services.dynamodb.model.DescribeTableResponse;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.KeySchemaElement;
import software.amazon.awssdk.services.dynamodb.model.KeyType;
import software.amazon.awssdk.services.dynamodb.model.ProvisionedThroughput;
import software.amazon.awssdk.services.dynamodb.model.ScalarAttributeType;
import software.amazon.awssdk.services.dynamodb.waiters.DynamoDbWaiter;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateTable {
    public static void main(String[] args) {
        final String usage = ""
```

```

Usage:
    <tableName> <key>

Where:
    tableName - The Amazon DynamoDB table to create (for example,
Music3).
    key - The key for the Amazon DynamoDB table (for example,
Artist).
    """;

if (args.length != 2) {
    System.out.println(usage);
    System.exit(1);
}

String tableName = args[0];
String key = args[1];
System.out.println("Creating an Amazon DynamoDB table " + tableName + " with
a simple primary key: " + key);
Region region = Region.US_EAST_1;
DynamoDbClient ddb = DynamoDbClient.builder()
    .region(region)
    .build();

String result = createTable(ddb, tableName, key);
System.out.println("New table is " + result);
ddb.close();
}

public static String createTable(DynamoDbClient ddb, String tableName, String
key) {
    DynamoDbWaiter dbWaiter = ddb.waiter();
    CreateTableRequest request = CreateTableRequest.builder()
        .attributeDefinitions(AttributeDefinition.builder()
            .attributeName(key)
            .attributeType(ScalarAttributeType.S)
            .build())
        .keySchema(KeySchemaElement.builder()
            .attributeName(key)
            .keyType(KeyType.HASH)
            .build())
        .provisionedThroughput(ProvisionedThroughput.builder()
            .readCapacityUnits(10L)

```

```
        .writeCapacityUnits(10L)
        .build()
    .tableName(tableName)
    .build();

String newTable;
try {
    CreateTableResponse response = ddb.createTable(request);
    DescribeTableRequest tableRequest = DescribeTableRequest.builder()
        .tableName(tableName)
        .build();

    // Wait until the Amazon DynamoDB table is created.
    WaiterResponse<DescribeTableResponse> waiterResponse =
dbWaiter.waitUntilTableExists(tableRequest);
    waiterResponse.matched().response().ifPresent(System.out::println);
    newTable = response.tableDescription().tableName();
    return newTable;

} catch (DynamoDbException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
return "";
}
}
```

- Per i dettagli sull'API, [CreateTable](#) consulta AWS SDK for Java 2.x API Reference.

## DeleteItem

Il seguente esempio di codice mostra come utilizzare `DeleteItem`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DeleteItemRequest;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import java.util.HashMap;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteItem {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <tableName> <key> <keyval>

                Where:
                tableName - The Amazon DynamoDB table to delete the item from
(for example, Music3).
                key - The key used in the Amazon DynamoDB table (for example,
Artist).\s
                keyval - The key value that represents the item to delete (for
example, Famous Band).
                """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String tableName = args[0];
        String key = args[1];
        String keyVal = args[2];
        System.out.format("Deleting item \"%s\" from %s\n", keyVal, tableName);
        Region region = Region.US_EAST_1;
        DynamoDbClient ddb = DynamoDbClient.builder()
                .region(region)
```

```
        .build();

        deleteDynamoDBItem(ddb, tableName, key, keyVal);
        ddb.close();
    }

    public static void deleteDynamoDBItem(DynamoDbClient ddb, String tableName,
String key, String keyVal) {
        HashMap<String, AttributeValue> keyToGet = new HashMap<>();
        keyToGet.put(key, AttributeValue.builder()
            .s(keyVal)
            .build());

        DeleteItemRequest deleteReq = DeleteItemRequest.builder()
            .tableName(tableName)
            .key(keyToGet)
            .build();

        try {
            ddb.deleteItem(deleteReq);
        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Per i dettagli sull'API, [DeleteItem](#) consulta AWS SDK for Java 2.x API Reference.

## DeleteTable

Il seguente esempio di codice mostra come utilizzare `DeleteTable`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.DeleteTableRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class DeleteTable {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <tableName>

            Where:
                tableName - The Amazon DynamoDB table to delete (for example,
Music3).

            **Warning** This program will delete the table that you specify!
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String tableName = args[0];
        System.out.format("Deleting the Amazon DynamoDB table %s...\n", tableName);
        Region region = Region.US_EAST_1;
        DynamoDbClient ddb = DynamoDbClient.builder()
            .region(region)
            .build();

        deleteDynamoDBTable(ddb, tableName);
        ddb.close();
    }
}
```

```
public static void deleteDynamoDBTable(DynamoDbClient ddb, String tableName) {
    DeleteTableRequest request = DeleteTableRequest.builder()
        .tableName(tableName)
        .build();

    try {
        ddb.deleteTable(request);
    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    System.out.println(tableName + " was successfully deleted!");
}
```

- Per i dettagli sull'API, [DeleteTable](#) consulta AWS SDK for Java 2.x API Reference.

## DescribeTable

Il seguente esempio di codice mostra come utilizzare `DescribeTable`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeDefinition;
import software.amazon.awssdk.services.dynamodb.model.DescribeTableRequest;
import
    software.amazon.awssdk.services.dynamodb.model.ProvisionedThroughputDescription;
import software.amazon.awssdk.services.dynamodb.model.TableDescription;
import java.util.List;
```



```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DescribeTable {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <tableName>

            Where:
                tableName - The Amazon DynamoDB table to get information about
(for example, Music3).
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String tableName = args[0];
        System.out.format("Getting description for %s\n\n", tableName);
        Region region = Region.US_EAST_1;
        DynamoDbClient ddb = DynamoDbClient.builder()
            .region(region)
            .build();

        describeDynamoDBTable(ddb, tableName);
        ddb.close();
    }

    public static void describeDynamoDBTable(DynamoDbClient ddb, String tableName) {
        DescribeTableRequest request = DescribeTableRequest.builder()
            .tableName(tableName)
            .build();

        try {
            TableDescription tableInfo = ddb.describeTable(request).table();
            if (tableInfo != null) {
```

```

        System.out.format("Table name   : %s\n", tableInfo.tableName());
        System.out.format("Table ARN    : %s\n", tableInfo.tableArn());
        System.out.format("Status      : %s\n", tableInfo.tableStatus());
        System.out.format("Item count  : %d\n", tableInfo.itemCount());
        System.out.format("Size (bytes): %d\n", tableInfo.tableSizeBytes());

        ProvisionedThroughputDescription throughputInfo =
tableInfo.provisionedThroughput();
        System.out.println("Throughput");
        System.out.format("  Read Capacity : %d\n",
throughputInfo.readCapacityUnits());
        System.out.format("  Write Capacity: %d\n",
throughputInfo.writeCapacityUnits());

        List<AttributeDefinition> attributes =
tableInfo.attributeDefinitions();
        System.out.println("Attributes");
        for (AttributeDefinition a : attributes) {
            System.out.format("  %s (%s)\n", a.attributeName(),
a.attributeType());
        }
    }

    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    System.out.println("\nDone!");
}
}

```

- Per i dettagli sull'API, [DescribeTable](#) consulta AWS SDK for Java 2.x API Reference.

## DescribeTimeToLive

Il seguente esempio di codice mostra come utilizzare `DescribeTimeToLive`.

SDK per Java 2.x

Descrivi la configurazione TTL su una tabella DynamoDB esistente.

```
import software.amazon.awssdk.regions.Region;
```

```
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.DescribeTimeToLiveRequest;
import software.amazon.awssdk.services.dynamodb.model.DescribeTimeToLiveResponse;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;

import java.util.Optional;

    final DescribeTimeToLiveRequest request =
DescribeTimeToLiveRequest.builder()
    .tableName(tableName)
    .build();
    try (DynamoDbClient ddb = DynamoDbClient.builder()
        .region(region)
        .build()) {
        final DescribeTimeToLiveResponse response =
ddb.describeTimeToLive(request);
        System.out.println(tableName + " description of time to live is "
            + response.toString());
    } catch (ResourceNotFoundException e) {
        System.err.format("Error: The Amazon DynamoDB table \"%s\" can't be
found.\n", tableName);
        System.exit(1);
    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    System.exit(0);
```

- Per i dettagli sull'API, consulta la sezione API [DescribeTimeToLive](#)Reference AWS SDK for Java 2.x .

## GetItem

Il seguente esempio di codice mostra come utilizzare `GetItem`.

## SDK per Java 2.x

 Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Ottiene un elemento da una tabella utilizzando DynamoDbClient.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.GetItemRequest;
import java.util.HashMap;
import java.util.Map;
import java.util.Set;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * To get an item from an Amazon DynamoDB table using the AWS SDK for Java V2,
 * its better practice to use the
 * Enhanced Client, see the EnhancedGetItem example.
 */
public class GetItem {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <tableName> <key> <keyVal>

                Where:
                tableName - The Amazon DynamoDB table from which an item is
retrieved (for example, Music3).\s
                key - The key used in the Amazon DynamoDB table (for example,
Artist).\s
    }
}
```

```

        keyVal - The key value that represents the item to get (for
example, Famous Band).
        """;

    if (args.length != 3) {
        System.out.println(usage);
        System.exit(1);
    }

    String tableName = args[0];
    String key = args[1];
    String keyVal = args[2];
    System.out.format("Retrieving item \"%s\" from \"%s\"\n", keyVal,
tableName);
    Region region = Region.US_EAST_1;
    DynamoDbClient ddb = DynamoDbClient.builder()
        .region(region)
        .build();

    getDynamoDBItem(ddb, tableName, key, keyVal);
    ddb.close();
}

public static void getDynamoDBItem(DynamoDbClient ddb, String tableName, String
key, String keyVal) {
    HashMap<String, AttributeValue> keyToGet = new HashMap<>();
    keyToGet.put(key, AttributeValue.builder()
        .s(keyVal)
        .build());

    GetItemRequest request = GetItemRequest.builder()
        .key(keyToGet)
        .tableName(tableName)
        .build();

    try {
        // If there is no matching item, GetItem does not return any data.
        Map<String, AttributeValue> returnedItem = ddb.getItem(request).item();
        if (returnedItem.isEmpty())
            System.out.format("No item found with the key %s!\n", key);
        else {
            Set<String> keys = returnedItem.keySet();
            System.out.println("Amazon DynamoDB table attributes: \n");
            for (String key1 : keys) {

```

```
        System.out.format("%s: %s\n", key1,
returnedItem.get(key1).toString());
    }
}

} catch (DynamoDbException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
}
```

- Per i dettagli sull'API, vedere [GetItem](#) in AWS SDK for Java 2.x API Reference.

## ListTables

Il seguente esempio di codice mostra come utilizzare `ListTables`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.ListTablesRequest;
import software.amazon.awssdk.services.dynamodb.model.ListTablesResponse;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
```

```
public class ListTables {
    public static void main(String[] args) {
        System.out.println("Listing your Amazon DynamoDB tables:\n");
        Region region = Region.US_EAST_1;
        DynamoDbClient ddb = DynamoDbClient.builder()
            .region(region)
            .build();
        listAllTables(ddb);
        ddb.close();
    }

    public static void listAllTables(DynamoDbClient ddb) {
        boolean moreTables = true;
        String lastName = null;

        while (moreTables) {
            try {
                ListTablesResponse response = null;
                if (lastName == null) {
                    ListTablesRequest request = ListTablesRequest.builder().build();
                    response = ddb.listTables(request);
                } else {
                    ListTablesRequest request = ListTablesRequest.builder()
                        .exclusiveStartTableName(lastName).build();
                    response = ddb.listTables(request);
                }

                List<String> tableNames = response.tableNames();
                if (tableNames.size() > 0) {
                    for (String curName : tableNames) {
                        System.out.format("* %s\n", curName);
                    }
                } else {
                    System.out.println("No tables found!");
                    System.exit(0);
                }

                lastName = response.lastEvaluatedTableName();
                if (lastName == null) {
                    moreTables = false;
                }
            } catch (DynamoDbException e) {
                System.err.println(e.getMessage());
            }
        }
    }
}
```

```
        System.exit(1);
    }
}
System.out.println("\nDone!");
}
```

- Per i dettagli sull'API, [ListTables](#) consulta AWS SDK for Java 2.x API Reference.

## PutItem

Il seguente esempio di codice mostra come utilizzare `PutItem`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Mette un elemento in una tabella utilizzando [DynamoDbClient](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.PutItemRequest;
import software.amazon.awssdk.services.dynamodb.model.PutItemResponse;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;
import java.util.HashMap;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * To place items into an Amazon DynamoDB table using the AWS SDK for Java V2,
```



```

* its better practice to use the
* Enhanced Client. See the EnhancedPutItem example.
*/
public class PutItem {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                    <tableName> <key> <keyVal> <albumtitle> <albumtitleval> <awards>
<awardsval> <Songtitle> <songtitleval>

                Where:
                    tableName - The Amazon DynamoDB table in which an item is placed
(for example, Music3).
                    key - The key used in the Amazon DynamoDB table (for example,
Artist).
                    keyval - The key value that represents the item to get (for
example, Famous Band).
                    albumTitle - The Album title (for example, AlbumTitle).
                    AlbumTitleValue - The name of the album (for example, Songs
About Life ).
                    Awards - The awards column (for example, Awards).
                    AwardVal - The value of the awards (for example, 10).
                    SongTitle - The song title (for example, SongTitle).
                    SongTitleVal - The value of the song title (for example, Happy
Day).

                **Warning** This program will place an item that you specify into a
table!

                """;

        if (args.length != 9) {
            System.out.println(usage);
            System.exit(1);
        }

        String tableName = args[0];
        String key = args[1];
        String keyVal = args[2];
        String albumTitle = args[3];
        String albumTitleValue = args[4];
        String awards = args[5];
        String awardVal = args[6];
        String songTitle = args[7];
        String songTitleVal = args[8];

```

```
Region region = Region.US_EAST_1;
DynamoDbClient ddb = DynamoDbClient.builder()
    .region(region)
    .build();

putItemInTable(ddb, tableName, key, keyVal, albumTitle, albumTitleValue,
awards, awardVal, songTitle,
    songTitleVal);
System.out.println("Done!");
ddb.close();
}

public static void putItemInTable(DynamoDbClient ddb,
    String tableName,
    String key,
    String keyVal,
    String albumTitle,
    String albumTitleValue,
    String awards,
    String awardVal,
    String songTitle,
    String songTitleVal) {

    HashMap<String, AttributeValue> itemValues = new HashMap<>();
    itemValues.put(key, AttributeValue.builder().s(keyVal).build());
    itemValues.put(songTitle, AttributeValue.builder().s(songTitleVal).build());
    itemValues.put(albumTitle,
AttributeValue.builder().s(albumTitleValue).build());
    itemValues.put(awards, AttributeValue.builder().s(awardVal).build());

    PutItemRequest request = PutItemRequest.builder()
        .tableName(tableName)
        .item(itemValues)
        .build();

    try {
        PutItemResponse response = ddb.putItem(request);
        System.out.println(tableName + " was successfully updated. The request
id is "
            + response.responseMetadata().requestId());
    } catch (ResourceNotFoundException e) {
```

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

- Per i dettagli sull'API, vedi [PutItem](#) in AWS SDK for Java 2.x API Reference.

## Query

Il seguente esempio di codice mostra come utilizzare `Query`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Interroga una tabella utilizzando [DynamoDbClient](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.QueryRequest;
import software.amazon.awssdk.services.dynamodb.model.QueryResponse;
import java.util.HashMap;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
```

```
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*
* To query items from an Amazon DynamoDB table using the AWS SDK for Java V2,
* its better practice to use the
* Enhanced Client. See the EnhancedQueryRecords example.
*/
public class Query {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <tableName> <partitionKeyName> <partitionKeyVal>

            Where:
                tableName - The Amazon DynamoDB table to put the item in (for
example, Music3).
                partitionKeyName - The partition key name of the Amazon DynamoDB
table (for example, Artist).
                partitionKeyVal - The value of the partition key that should
match (for example, Famous Band).
                """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String tableName = args[0];
        String partitionKeyName = args[1];
        String partitionKeyVal = args[2];

        // For more information about an alias, see:
        // https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/
Expressions.ExpressionAttributeNames.html
        String partitionAlias = "#a";

        System.out.format("Querying %s", tableName);
        System.out.println("");
        Region region = Region.US_EAST_1;
        DynamoDbClient ddb = DynamoDbClient.builder()
            .region(region)
            .build();
```

```

        int count = queryTable(ddb, tableName, partitionKeyName, partitionKeyVal,
partitionAlias);
        System.out.println("There were " + count + " record(s) returned");
        ddb.close();
    }

    public static int queryTable(DynamoDbClient ddb, String tableName, String
partitionKeyName, String partitionKeyVal,
        String partitionAlias) {
        // Set up an alias for the partition key name in case it's a reserved word.
        HashMap<String, String> attrNameAlias = new HashMap<String, String>();
        attrNameAlias.put(partitionAlias, partitionKeyName);

        // Set up mapping of the partition name with the value.
        HashMap<String, AttributeValue> attrValues = new HashMap<>();
        attrValues.put(":" + partitionKeyName, AttributeValue.builder()
            .s(partitionKeyVal)
            .build());

        QueryRequest queryReq = QueryRequest.builder()
            .tableName(tableName)
            .keyConditionExpression(partitionAlias + " = :" + partitionKeyName)
            .expressionAttributeNames(attrNameAlias)
            .expressionAttributeValues(attrValues)
            .build();

        try {
            QueryResponse response = ddb.query(queryReq);
            return response.count();

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        return -1;
    }
}

```

Esegue query su una tabella utilizzando `DynamoDbClient` e un indice secondario.

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;

```

```
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.QueryRequest;
import software.amazon.awssdk.services.dynamodb.model.QueryResponse;
import java.util.HashMap;
import java.util.Map;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * Create the Movies table by running the Scenario example and loading the Movie
 * data from the JSON file. Next create a secondary
 * index for the Movies table that uses only the year column. Name the index
 * year-index. For more information, see:
 *
 * https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/GSI.html
 */
public class QueryItemsUsingIndex {
    public static void main(String[] args) {
        String tableName = "Movies";
        Region region = Region.US_EAST_1;
        DynamoDbClient ddb = DynamoDbClient.builder()
            .region(region)
            .build();

        queryIndex(ddb, tableName);
        ddb.close();
    }

    public static void queryIndex(DynamoDbClient ddb, String tableName) {
        try {
            Map<String, String> expressionAttributesNames = new HashMap<>();
            expressionAttributesNames.put("#year", "year");
            Map<String, AttributeValue> expressionAttributeValues = new HashMap<>();
            expressionAttributeValues.put(":yearValue",
                AttributeValue.builder().n("2013").build());

            QueryRequest request = QueryRequest.builder()
                .tableName(tableName)
```

```

        .indexName("year-index")
        .keyConditionExpression("#year = :yearValue")
        .expressionAttributeNames(expressionAttributeNames)
        .expressionAttributeValues(expressionAttributeValues)
        .build();

    System.out.println("=== Movie Titles ===");
    QueryResponse response = ddb.query(request);
    response.items()
        .forEach(movie -> System.out.println(movie.get("title").s()));

    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
}
}

```

- Per ulteriori informazioni sulle API, consulta [Query](#) nella Documentazione di riferimento delle API AWS SDK for Java 2.x .

## Scan

Il seguente esempio di codice mostra come usare Scan.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Esegue la scansione di una tabella Amazon DynamoDB utilizzando [DynamoDbClient](#).

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.ScanRequest;
import software.amazon.awssdk.services.dynamodb.model.ScanResponse;

```

```
import java.util.Map;
import java.util.Set;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * To scan items from an Amazon DynamoDB table using the AWS SDK for Java V2,
 * its better practice to use the
 * Enhanced Client, See the EnhancedScanRecords example.
 */

public class DynamoDBScanItems {
    public static void main(String[] args) {

        final String usage = ""

            Usage:
                <tableName>

            Where:
                tableName - The Amazon DynamoDB table to get information from
(for example, Music3).
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String tableName = args[0];
        Region region = Region.US_EAST_1;
        DynamoDbClient ddb = DynamoDbClient.builder()
            .region(region)
            .build();

        scanItems(ddb, tableName);
        ddb.close();
    }
}
```



```
public static void scanItems(DynamoDbClient ddb, String tableName) {
    try {
        ScanRequest scanRequest = ScanRequest.builder()
            .tableName(tableName)
            .build();

        ScanResponse response = ddb.scan(scanRequest);
        for (Map<String, AttributeValue> item : response.items()) {
            Set<String> keys = item.keySet();
            for (String key : keys) {
                System.out.println("The key name is " + key + "\n");
                System.out.println("The value is " + item.get(key).s());
            }
        }

    } catch (DynamoDbException e) {
        e.printStackTrace();
        System.exit(1);
    }
}
```

- Per informazioni dettagliate sulle API, consulta [Scan](#) nella Documentazione di riferimento per le API AWS SDK for Java 2.x .

## UpdateItem

Il seguente esempio di codice mostra come usare UpdateItem

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Aggiorna un elemento in una tabella utilizzando [DynamoDbClient](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
```

```
import software.amazon.awssdk.services.dynamodb.model.AttributeAction;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.AttributeValueUpdate;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemRequest;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import java.util.HashMap;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * To update an Amazon DynamoDB table using the AWS SDK for Java V2, its better
 * practice to use the
 * Enhanced Client, See the EnhancedModifyItem example.
 */
public class UpdateItem {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <tableName> <key> <keyVal> <name> <updateVal>

            Where:
                tableName - The Amazon DynamoDB table (for example, Music3).
                key - The name of the key in the table (for example, Artist).
                keyVal - The value of the key (for example, Famous Band).
                name - The name of the column where the value is updated (for
example, Awards).
                updateVal - The value used to update an item (for example, 14).
            Example:
                UpdateItem Music3 Artist Famous Band Awards 14
            """;

        if (args.length != 5) {
            System.out.println(usage);
            System.exit(1);
        }

        String tableName = args[0];
        String key = args[1];
```

```
String keyVal = args[2];
String name = args[3];
String updateVal = args[4];

Region region = Region.US_EAST_1;
DynamoDbClient ddb = DynamoDbClient.builder()
    .region(region)
    .build();
updateTableItem(ddb, tableName, key, keyVal, name, updateVal);
ddb.close();
}

public static void updateTableItem(DynamoDbClient ddb,
    String tableName,
    String key,
    String keyVal,
    String name,
    String updateVal) {

    HashMap<String, AttributeValue> itemKey = new HashMap<>();
    itemKey.put(key, AttributeValue.builder()
        .s(keyVal)
        .build());

    HashMap<String, AttributeValueUpdate> updatedValues = new HashMap<>();
    updatedValues.put(name, AttributeValueUpdate.builder()
        .value(AttributeValue.builder().s(updateVal).build())
        .action(AttributeAction.PUT)
        .build());

    UpdateItemRequest request = UpdateItemRequest.builder()
        .tableName(tableName)
        .key(itemKey)
        .attributeUpdates(updatedValues)
        .build();

    try {
        ddb.updateItem(request);
    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    System.out.println("The Amazon DynamoDB table was updated!");
}
```

```
}
```

- Per i dettagli sull'API, vedi [UpdateItem](#) in AWS SDK for Java 2.x API Reference.

## UpdateTimeToLive

Il seguente esempio di codice mostra come utilizzare `UpdateTimeToLive`.

### SDK per Java 2.x

Abilita TTL su una tabella DynamoDB esistente.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;
import software.amazon.awssdk.services.dynamodb.model.TimeToLiveSpecification;
import software.amazon.awssdk.services.dynamodb.model.UpdateTimeToLiveRequest;
import software.amazon.awssdk.services.dynamodb.model.UpdateTimeToLiveResponse;

import java.util.Optional;

    final TimeToLiveSpecification ttlSpecification =
TimeToLiveSpecification.builder()
    .attributeName(ttlAttributeName)
    .enabled(true)
    .build();
    final UpdateTimeToLiveRequest request = UpdateTimeToLiveRequest.builder()
    .tableName(tableName)
    .timeToLiveSpecification(ttlSpecification)
    .build();
    try (DynamoDbClient ddb = DynamoDbClient.builder()
    .region(region)
    .build()) {
        final UpdateTimeToLiveResponse response =
ddb.updateTimeToLive(request);
        System.out.println(tableName + " had its TTL successfully updated.
The request id is "
            + response.responseMetadata().requestId());
    } catch (ResourceNotFoundException e) {
        System.err.format("Error: The Amazon DynamoDB table \"%s\" can't be
found.\n", tableName);
```

```

        System.exit(1);
    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    System.out.println("Done!");

```

Disabilita il TTL su una tabella DynamoDB esistente.

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;
import software.amazon.awssdk.services.dynamodb.model.TimeToLiveSpecification;
import software.amazon.awssdk.services.dynamodb.model.UpdateTimeToLiveRequest;
import software.amazon.awssdk.services.dynamodb.model.UpdateTimeToLiveResponse;

import java.util.Optional;

    final Region region = Optional.ofNullable(args[2]).isEmpty() ?
Region.US_EAST_1 : Region.of(args[2]);
    final TimeToLiveSpecification ttlSpecification =
TimeToLiveSpecification.builder()
        .attributeName(ttlAttributeName)
        .enabled(false)
        .build();
    final UpdateTimeToLiveRequest request = UpdateTimeToLiveRequest.builder()
        .tableName(tableName)
        .timeToLiveSpecification(ttlSpecification)
        .build();
    try (DynamoDbClient ddb = DynamoDbClient.builder()
        .region(region)
        .build()) {
        final UpdateTimeToLiveResponse response = ddb.updateTimeToLive(request);
        System.out.println(tableName + " had its TTL successfully updated. The
request id is "
            + response.responseMetadata().requestId());
    } catch (ResourceNotFoundException e) {
        System.err.format("Error: The Amazon DynamoDB table \"%s\" can't be
found.\n", tableName);
        System.exit(1);
    } catch (DynamoDbException e) {

```

```

        System.err.println(e.getMessage());
        System.exit(1);
    }
    System.out.println("Done!");

```

- Per i dettagli sull'API, consulta la sezione API [UpdateTimeToLive](#) Reference AWS SDK for Java 2.x.

## Scenari

### Aggiorna in modo condizionale il TTL di un elemento

Il seguente esempio di codice mostra come aggiornare in modo condizionale il TTL di un elemento.

#### SDK per Java 2.x

```

// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package com.amazon.samplelib.ttl;

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemRequest;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemResponse;
import software.amazon.awssdk.utils.ImmutableMap;

import java.util.Map;
import java.util.Optional;

public class UpdateTTLConditional {
    public static void main(String[] args) {
        final String usage = ""
            Usage:
                <tableName> <primaryKey> <sortKey> <newTtlAttribute> <region>
            Where:
                tableName - The Amazon DynamoDB table being queried.
                primaryKey - The name of the primary key. Also known as the hash
or partition key.

```

```

        sortKey - The name of the sort key. Also known as the range
attribute.
        newTtlAttribute - New attribute name (as part of the update
command)
        region (optional) - The AWS region that the Amazon DynamoDB
table is located in. (Default: us-east-1)
        """;
    // Optional "region" parameter - if args list length is NOT 3 or 4, short-
circuit exit.
    if (!(args.length == 4 || args.length == 5)) {
        System.out.println(usage);
        System.exit(1);
    }
    final String tableName = args[0];
    final String primaryKey = args[1];
    final String sortKey = args[2];
    final String newTtlAttribute = args[3];
    Region region = Optional.ofNullable(args[4]).isEmpty() ? Region.US_EAST_1 :
Region.of(args[4]);

    // Get current time in epoch second format
    final long currentTime = System.currentTimeMillis() / 1000;
    // Calculate expiration time 90 days from now in epoch second format
    final long expireDate = currentTime + (90 * 24 * 60 * 60);
    // An expression that defines one or more attributes to be updated, the
action to be performed on them, and new values for them.
    final String updateExpression = "SET newTtlAttribute = :val1";
    // A condition that must be satisfied in order for a conditional update to
succeed.
    final String conditionExpression = "expireAt > :val2";

    final ImmutableMap<String, AttributeValue> keyMap =
        ImmutableMap.of("primaryKey", AttributeValue.fromS(primaryKey),
            "sortKey", AttributeValue.fromS(sortKey));
    final Map<String, AttributeValue> expressionAttributeValues =
ImmutableMap.of(
        ":val1", AttributeValue.builder().s(newTtlAttribute).build(),
        ":val2",
AttributeValue.builder().s(String.valueOf(expireDate)).build()
    );

    final UpdateItemRequest request = UpdateItemRequest.builder()
        .tableName(tableName)
        .key(keyMap)

```

```

        .updateExpression(updateExpression)
        .conditionExpression(conditionExpression)
        .expressionAttributeValues(expressionAttributeValues)
        .build();
    try (DynamoDbClient ddb = DynamoDbClient.builder()
        .region(region)
        .build()) {
        final UpdateItemResponse response = ddb.updateItem(request);
        System.out.println(tableName + " UpdateItem operation with conditional
TTL successful. Request id is "
            + response.responseMetadata().requestId());
    } catch (ResourceNotFoundException e) {
        System.err.format("Error: The Amazon DynamoDB table \"%s\" can't be
found.\n", tableName);
        System.exit(1);
    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    System.exit(0);
}
}
}

```

- Per i dettagli sull'API, consulta la sezione AWS SDK for Java 2.x API [UpdateItem](#) Reference.

## Crea un elemento con un TTL

Il seguente esempio di codice mostra come creare un elemento con TTL.

### SDK per Java 2.x

```

// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0

package com.amazon.samplelib.ttl;

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.PutItemRequest;
import software.amazon.awssdk.services.dynamodb.model.PutItemResponse;

```



```
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;
import software.amazon.awssdk.utils.ImmutableMap;

import java.io.Serializable;
import java.util.Map;
import java.util.Optional;

public class CreateTTL {
    public static void main(String[] args) {
        final String usage = ""
            Usage:
                <tableName> <primaryKey> <sortKey> <region>
            Where:
                tableName - The Amazon DynamoDB table being queried.
                primaryKey - The name of the primary key. Also known as the hash
or partition key.
                sortKey - The name of the sort key. Also known as the range
attribute.
                region (optional) - The AWS region that the Amazon DynamoDB
table is located in. (Default: us-east-1)
            """;
        // Optional "region" parameter - if args list length is NOT 3 or 4, short-
circuit exit.
        if (!(args.length == 3 || args.length == 4)) {
            System.out.println(usage);
            System.exit(1);
        }

        String tableName = args[0];
        String primaryKey = args[1];
        String sortKey = args[2];
        Region region = Optional.ofNullable(args[3]).isEmpty() ? Region.US_EAST_1 :
Region.of(args[3]);

        // Get current time in epoch second format
        final long createDate = System.currentTimeMillis() / 1000;

        // Calculate expiration time 90 days from now in epoch second format
        final long expireDate = createDate + (90 * 24 * 60 * 60);

        final ImmutableMap<String, ? extends Serializable> itemMap =
            ImmutableMap.of("primaryKey", primaryKey,
                "sortKey", sortKey,
                "creationDate", createDate,
```

```

        "expireAt", expireDate);
    final PutItemRequest request = PutItemRequest.builder()
        .tableName(tableName)
        .item((Map<String, AttributeValue>) itemMap)
        .build();
    try (DynamoDbClient ddb = DynamoDbClient.builder()
        .region(region)
        .build()) {
        final PutItemResponse response = ddb.putItem(request);
        System.out.println(tableName + " PutItem operation with TTL successful.
Request id is "
            + response.responseMetadata().requestId());
    } catch (ResourceNotFoundException e) {
        System.err.format("Error: The Amazon DynamoDB table \"%s\" can't be
found.\n", tableName);
        System.exit(1);
    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    System.exit(0);
}
}

```

- Per i dettagli sull'API, consulta [PutItem AWS SDK for Java 2.x API Reference](#).

Nozioni di base sull'utilizzo di tabelle, elementi e query

L'esempio di codice seguente mostra come:

- Crea una tabella in grado di contenere i dati del filmato.
- Inserisci, ottieni e aggiorna un singolo filmato nella tabella.
- Scrivi i dati del filmato nella tabella da un file JSON di esempio.
- Esegui una query sui filmati che sono stati rilasciati in un dato anno.
- Cerca i filmati che sono stati distribuiti in diversi anni.
- Elimina un filmato dalla tabella, quindi elimina la tabella.

## SDK per Java 2.x

 Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

## Creazione di una tabella DynamoDB

```
// Create a table with a Sort key.
public static void createTable(DynamoDbClient ddb, String tableName) {
    DynamoDbWaiter dbWaiter = ddb.waiter();
    ArrayList<AttributeDefinition> attributeDefinitions = new ArrayList<>();

    // Define attributes.
    attributeDefinitions.add(AttributeDefinition.builder()
        .attributeName("year")
        .attributeType("N")
        .build());

    attributeDefinitions.add(AttributeDefinition.builder()
        .attributeName("title")
        .attributeType("S")
        .build());

    ArrayList<KeySchemaElement> tableKey = new ArrayList<>();
    KeySchemaElement key = KeySchemaElement.builder()
        .attributeName("year")
        .keyType(KeyType.HASH)
        .build();

    KeySchemaElement key2 = KeySchemaElement.builder()
        .attributeName("title")
        .keyType(KeyType.RANGE)
        .build();

    // Add KeySchemaElement objects to the list.
    tableKey.add(key);
    tableKey.add(key2);

    CreateTableRequest request = CreateTableRequest.builder()
        .keySchema(tableKey)
```

```

        .provisionedThroughput(ProvisionedThroughput.builder()
            .readCapacityUnits(10L)
            .writeCapacityUnits(10L)
            .build())
        .attributeDefinitions(attributeDefinitions)
        .tableName(tableName)
        .build();

    try {
        CreateTableResponse response = ddb.createTable(request);
        DescribeTableRequest tableRequest = DescribeTableRequest.builder()
            .tableName(tableName)
            .build();

        // Wait until the Amazon DynamoDB table is created.
        WaiterResponse<DescribeTableResponse> waiterResponse =
dbWaiter.waitUntilTableExists(tableRequest);
        waiterResponse.matched().response().ifPresent(System.out::println);
        String newTable = response.tableDescription().tableName();
        System.out.println("The " + newTable + " was successfully created.");

    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

```

Crea una funzione helper per scaricare ed estrarre il file JSON di esempio.

```

// Load data into the table.
public static void loadData(DynamoDbClient ddb, String tableName, String
fileName) throws IOException {
    DynamoDbEnhancedClient enhancedClient = DynamoDbEnhancedClient.builder()
        .dynamoDbClient(ddb)
        .build();

    DynamoDbTable<Movies> mappedTable = enhancedClient.table("Movies",
TableSchema.fromBean(Movies.class));
    JsonParser parser = new JsonFactory().createParser(new File(fileName));
    com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
    Iterator<JsonNode> iter = rootNode.iterator();

```

```
ObjectNode currentNode;
int t = 0;
while (iter.hasNext()) {
    // Only add 200 Movies to the table.
    if (t == 200)
        break;
    currentNode = (ObjectNode) iter.next();

    int year = currentNode.path("year").asInt();
    String title = currentNode.path("title").asText();
    String info = currentNode.path("info").toString();

    Movies movies = new Movies();
    movies.setYear(year);
    movies.setTitle(title);
    movies.setInfo(info);

    // Put the data into the Amazon DynamoDB Movie table.
    mappedTable.putItem(movies);
    t++;
}
}
```

## Ottieni un elemento da una tabella

```
public static void getItem(DynamoDbClient ddb) {

    HashMap<String, AttributeValue> keyToGet = new HashMap<>();
    keyToGet.put("year", AttributeValue.builder()
        .n("1933")
        .build());

    keyToGet.put("title", AttributeValue.builder()
        .s("King Kong")
        .build());

    GetItemRequest request = GetItemRequest.builder()
        .key(keyToGet)
        .tableName("Movies")
        .build();

    try {
```

```
Map<String, AttributeValue> returnedItem = ddb.getItem(request).item();

if (returnedItem != null) {
    Set<String> keys = returnedItem.keySet();
    System.out.println("Amazon DynamoDB table attributes: \n");

    for (String key1 : keys) {
        System.out.format("%s: %s\n", key1,
returnedItem.get(key1).toString());
    }
} else {
    System.out.format("No item found with the key %s!\n", "year");
}

} catch (DynamoDbException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
}
```

## Esempio completo.

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * This Java example performs these tasks:
 *
 * 1. Creates the Amazon DynamoDB Movie table with partition and sort key.
 * 2. Puts data into the Amazon DynamoDB table from a JSON document using the
 * Enhanced client.
 * 3. Gets data from the Movie table.
 * 4. Adds a new item.
 * 5. Updates an item.
 * 6. Uses a Scan to query items using the Enhanced client.
 * 7. Queries all items where the year is 2013 using the Enhanced Client.
 * 8. Deletes the table.
 */
```

```
public class Scenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    public static void main(String[] args) throws IOException {
        final String usage = ""

            Usage:
                <fileName>

            Where:
                fileName - The path to the moviedata.json file that you can
download from the Amazon DynamoDB Developer Guide.
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String tableName = "Movies";
        String fileName = args[0];
        Region region = Region.US_EAST_1;
        DynamoDbClient ddb = DynamoDbClient.builder()
            .region(region)
            .build();

        System.out.println(DASHES);
        System.out.println("Welcome to the Amazon DynamoDB example scenario.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println(
            "1. Creating an Amazon DynamoDB table named Movies with a key named
year and a sort key named title.");
        createTable(ddb, tableName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("2. Loading data into the Amazon DynamoDB table.");
        loadData(ddb, tableName, fileName);
        System.out.println(DASHES);

        System.out.println(DASHES);
    }
}
```

```
        System.out.println("3. Getting data from the Movie table.");
        getItem(ddb);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. Putting a record into the Amazon DynamoDB table.");
        putRecord(ddb);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. Updating a record.");
        updateTableItem(ddb, tableName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6. Scanning the Amazon DynamoDB table.");
        scanMovies(ddb, tableName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Querying the Movies released in 2013.");
        queryTable(ddb);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("8. Deleting the Amazon DynamoDB table.");
        deleteDynamoDBTable(ddb, tableName);
        System.out.println(DASHES);

        ddb.close();
    }

    // Create a table with a Sort key.
    public static void createTable(DynamoDbClient ddb, String tableName) {
        DynamoDbWaiter dbWaiter = ddb.waiter();
        ArrayList<AttributeDefinition> attributeDefinitions = new ArrayList<>();

        // Define attributes.
        attributeDefinitions.add(AttributeDefinition.builder()
            .attributeName("year")
            .attributeType("N")
            .build());

        attributeDefinitions.add(AttributeDefinition.builder()
```



```
        .attributeName("title")
        .attributeType("S")
        .build());

ArrayList<KeySchemaElement> tableKey = new ArrayList<>();
KeySchemaElement key = KeySchemaElement.builder()
    .attributeName("year")
    .keyType(KeyType.HASH)
    .build();

KeySchemaElement key2 = KeySchemaElement.builder()
    .attributeName("title")
    .keyType(KeyType.RANGE)
    .build();

// Add KeySchemaElement objects to the list.
tableKey.add(key);
tableKey.add(key2);

CreateTableRequest request = CreateTableRequest.builder()
    .keySchema(tableKey)
    .provisionedThroughput(ProvisionedThroughput.builder()
        .readCapacityUnits(10L)
        .writeCapacityUnits(10L)
        .build())
    .attributeDefinitions(attributeDefinitions)
    .tableName(tableName)
    .build();

try {
    CreateTableResponse response = ddb.createTable(request);
    DescribeTableRequest tableRequest = DescribeTableRequest.builder()
        .tableName(tableName)
        .build();

    // Wait until the Amazon DynamoDB table is created.
    WaiterResponse<DescribeTableResponse> waiterResponse =
dbWaiter.waitUntilTableExists(tableRequest);
    waiterResponse.matched().response().ifPresent(System.out::println);
    String newTable = response.tableDescription().tableName();
    System.out.println("The " + newTable + " was successfully created.");

} catch (DynamoDbException e) {
    System.err.println(e.getMessage());
}
```

```
        System.exit(1);
    }
}

// Query the table.
public static void queryTable(DynamoDbClient ddb) {
    try {
        DynamoDbEnhancedClient enhancedClient = DynamoDbEnhancedClient.builder()
            .dynamoDbClient(ddb)
            .build();

        DynamoDbTable<Movies> custTable = enhancedClient.table("Movies",
TableSchema.fromBean(Movies.class));
        QueryConditional queryConditional = QueryConditional
            .keyEqualTo(Key.builder()
                .partitionValue(2013)
                .build());

        // Get items in the table and write out the ID value.
        Iterator<Movies> results =
custTable.query(queryConditional).items().iterator();
        String result = "";

        while (results.hasNext()) {
            Movies rec = results.next();
            System.out.println("The title of the movie is " + rec.getTitle());
            System.out.println("The movie information is " + rec.getInfo());
        }

    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

// Scan the table.
public static void scanMovies(DynamoDbClient ddb, String tableName) {
    System.out.println("***** Scanning all movies.\n");
    try {
        DynamoDbEnhancedClient enhancedClient = DynamoDbEnhancedClient.builder()
            .dynamoDbClient(ddb)
            .build();
```

```

        DynamoDbTable<Movies> custTable = enhancedClient.table("Movies",
TableSchema.fromBean(Movies.class));
        Iterator<Movies> results = custTable.scan().items().iterator();
        while (results.hasNext()) {
            Movies rec = results.next();
            System.out.println("The movie title is " + rec.getTitle());
            System.out.println("The movie year is " + rec.getYear());
        }

    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

// Load data into the table.
public static void loadData(DynamoDbClient ddb, String tableName, String
fileName) throws IOException {
    DynamoDbEnhancedClient enhancedClient = DynamoDbEnhancedClient.builder()
        .dynamoDbClient(ddb)
        .build();

    DynamoDbTable<Movies> mappedTable = enhancedClient.table("Movies",
TableSchema.fromBean(Movies.class));
    JsonParser parser = new JsonFactory().createParser(new File(fileName));
    com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
    Iterator<JsonNode> iter = rootNode.iterator();
    ObjectNode currentNode;
    int t = 0;
    while (iter.hasNext()) {
        // Only add 200 Movies to the table.
        if (t == 200)
            break;
        currentNode = (ObjectNode) iter.next();

        int year = currentNode.path("year").asInt();
        String title = currentNode.path("title").asText();
        String info = currentNode.path("info").toString();

        Movies movies = new Movies();
        movies.setYear(year);
        movies.setTitle(title);
        movies.setInfo(info);
    }
}

```

```

        // Put the data into the Amazon DynamoDB Movie table.
        mappedTable.putItem(movies);
        t++;
    }
}

// Update the record to include show only directors.
public static void updateTableItem(DynamoDbClient ddb, String tableName) {
    HashMap<String, AttributeValue> itemKey = new HashMap<>();
    itemKey.put("year", AttributeValue.builder().n("1933").build());
    itemKey.put("title", AttributeValue.builder().s("King Kong").build());

    HashMap<String, AttributeValueUpdate> updatedValues = new HashMap<>();
    updatedValues.put("info", AttributeValueUpdate.builder()
        .value(AttributeValue.builder().s("{\\"directors\\":[\\\"Merian C.
Cooper\\\",\\\"Ernest B. Schoedsack\\\"]}")
        .build())
        .action(AttributeAction.PUT)
        .build());

    UpdateItemRequest request = UpdateItemRequest.builder()
        .tableName(tableName)
        .key(itemKey)
        .attributeUpdates(updatedValues)
        .build();

    try {
        ddb.updateItem(request);
    } catch (ResourceNotFoundException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }

    System.out.println("Item was updated!");
}

public static void deleteDynamoDBTable(DynamoDbClient ddb, String tableName) {
    DeleteTableRequest request = DeleteTableRequest.builder()
        .tableName(tableName)
        .build();
}

```

```
    try {
        ddb.deleteTable(request);

    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    System.out.println(tableName + " was successfully deleted!");
}

public static void putRecord(DynamoDbClient ddb) {
    try {
        DynamoDbEnhancedClient enhancedClient = DynamoDbEnhancedClient.builder()
            .dynamoDbClient(ddb)
            .build();

        DynamoDbTable<Movies> table = enhancedClient.table("Movies",
TableSchema.fromBean(Movies.class));

        // Populate the Table.
        Movies record = new Movies();
        record.setYear(2020);
        record.setTitle("My Movie2");
        record.setInfo("no info");
        table.putItem(record);

    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    System.out.println("Added a new movie to the table.");
}

public static void getItem(DynamoDbClient ddb) {

    HashMap<String, AttributeValue> keyToGet = new HashMap<>();
    keyToGet.put("year", AttributeValue.builder()
        .n("1933")
        .build());

    keyToGet.put("title", AttributeValue.builder()
        .s("King Kong")
        .build());
```

```
    GetItemRequest request = GetItemRequest.builder()
        .key(keyToGet)
        .tableName("Movies")
        .build();

    try {
        Map<String, AttributeValue> returnedItem = ddb.getItem(request).item();

        if (returnedItem != null) {
            Set<String> keys = returnedItem.keySet();
            System.out.println("Amazon DynamoDB table attributes: \n");

            for (String key1 : keys) {
                System.out.format("%s: %s\n", key1,
returnedItem.get(key1).toString());
            }
        } else {
            System.out.format("No item found with the key %s!\n", "year");
        }

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella Documentazione di riferimento delle API AWS SDK for Java 2.x .

- [BatchWriteItem](#)
- [CreateTable](#)
- [DeleteItem](#)
- [DeleteTable](#)
- [DescribeTable](#)
- [GetItem](#)
- [PutItem](#)
- [Query](#)


- [Scan](#)
- [UpdateItem](#)

Esecuzione di una query su una tabella mediante batch di istruzioni PartiQL

L'esempio di codice seguente mostra come:

- Ricezione di un batch di elementi mediante più istruzioni SELECT.
- Aggiunta di un batch di articoli eseguendo più istruzioni INSERT.
- Aggiornamento di un batch di elementi mediante più istruzioni UPDATE.
- Eliminazione di un batch di elementi mediante più istruzioni DELETE.

SDK per Java 2.x

 Note

C'è dell'altro GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public class ScenarioPartiQLBatch {
    public static void main(String[] args) throws IOException {
        String tableName = "MoviesPartiQBatch";
        Region region = Region.US_EAST_1;
        DynamoDbClient ddb = DynamoDbClient.builder()
            .region(region)
            .build();

        System.out.println("***** Creating an Amazon DynamoDB table named
" + tableName
            + " with a key named year and a sort key named
title.");
        createTable(ddb, tableName);

        System.out.println("***** Adding multiple records into the " +
tableName
            + " table using a batch command.");
        putRecordBatch(ddb);
    }
}
```

```
        System.out.println("***** Updating multiple records using a batch
command.");
        updateTableItemBatch(ddb);

        System.out.println("***** Deleting multiple records using a batch
command.");
        deleteItemBatch(ddb);

        System.out.println("***** Deleting the Amazon DynamoDB table.");
        deleteDynamoDBTable(ddb, tableName);
        ddb.close();
    }

    public static void createTable(DynamoDbClient ddb, String tableName) {
        DynamoDbWaiter dbWaiter = ddb.waiter();
        ArrayList<AttributeDefinition> attributeDefinitions = new
ArrayList<>();

        // Define attributes.
        attributeDefinitions.add(AttributeDefinition.builder()
            .attributeName("year")
            .attributeType("N")
            .build());

        attributeDefinitions.add(AttributeDefinition.builder()
            .attributeName("title")
            .attributeType("S")
            .build());

        ArrayList<KeySchemaElement> tableKey = new ArrayList<>();
        KeySchemaElement key = KeySchemaElement.builder()
            .attributeName("year")
            .keyType(KeyType.HASH)
            .build();

        KeySchemaElement key2 = KeySchemaElement.builder()
            .attributeName("title")
            .keyType(KeyType.RANGE) // Sort
            .build();

        // Add KeySchemaElement objects to the list.
        tableKey.add(key);
        tableKey.add(key2);
    }
}
```



```

        CreateTableRequest request = CreateTableRequest.builder()
            .keySchema(tableKey)

        .provisionedThroughput(ProvisionedThroughput.builder()
            .readCapacityUnits(new Long(10))
            .writeCapacityUnits(new Long(10))
            .build())
            .attributeDefinitions(attributeDefinitions)
            .tableName(tableName)
            .build();

        try {
            CreateTableResponse response = ddb.createTable(request);
            DescribeTableRequest tableRequest =
DescribeTableRequest.builder()
                .tableName(tableName)
                .build();

            // Wait until the Amazon DynamoDB table is created.
            WaiterResponse<DescribeTableResponse> waiterResponse =
dbWaiter
                .waitUntilTableExists(tableRequest);

            waiterResponse.matched().response().ifPresent(System.out::println);
            String newTable = response.tableDescription().tableName();
            System.out.println("The " + newTable + " was successfully
created.");

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void putRecordBatch(DynamoDbClient ddb) {
        String sqlStatement = "INSERT INTO MoviesPartiQBatch VALUE
{'year':?, 'title' : ?, 'info' : ?}";
        try {
            // Create three movies to add to the Amazon DynamoDB table.
            // Set data for Movie 1.
            List<AttributeValue> parameters = new ArrayList<>();

            AttributeValue att1 = AttributeValue.builder()
                .n(String.valueOf("2022"))

```

```
        .build();

        AttributeValue att2 = AttributeValue.builder()
            .s("My Movie 1")
            .build();

        AttributeValue att3 = AttributeValue.builder()
            .s("No Information")
            .build();

        parameters.add(att1);
        parameters.add(att2);
        parameters.add(att3);

        BatchStatementRequest statementRequestMovie1 =
BatchStatementRequest.builder()
            .statement(sqlStatement)
            .parameters(parameters)
            .build();

        // Set data for Movie 2.
        List<AttributeValue> parametersMovie2 = new ArrayList<>();
        AttributeValue attMovie2 = AttributeValue.builder()
            .n(String.valueOf("2022"))
            .build();

        AttributeValue attMovie2A = AttributeValue.builder()
            .s("My Movie 2")
            .build();

        AttributeValue attMovie2B = AttributeValue.builder()
            .s("No Information")
            .build();

        parametersMovie2.add(attMovie2);
        parametersMovie2.add(attMovie2A);
        parametersMovie2.add(attMovie2B);

        BatchStatementRequest statementRequestMovie2 =
BatchStatementRequest.builder()
            .statement(sqlStatement)
            .parameters(parametersMovie2)
            .build();
```

```
// Set data for Movie 3.
List<AttributeValue> parametersMovie3 = new ArrayList<>();
AttributeValue attMovie3 = AttributeValue.builder()
    .n(String.valueOf("2022"))
    .build();

AttributeValue attMovie3A = AttributeValue.builder()
    .s("My Movie 3")
    .build();

AttributeValue attMovie3B = AttributeValue.builder()
    .s("No Information")
    .build();

parametersMovie3.add(attMovie3);
parametersMovie3.add(attMovie3A);
parametersMovie3.add(attMovie3B);

BatchStatementRequest statementRequestMovie3 =
BatchStatementRequest.builder()
    .statement(sqlStatement)
    .parameters(parametersMovie3)
    .build();

// Add all three movies to the list.
List<BatchStatementRequest> myBatchStatementList = new
ArrayList<>();

myBatchStatementList.add(statementRequestMovie1);
myBatchStatementList.add(statementRequestMovie2);
myBatchStatementList.add(statementRequestMovie3);

BatchExecuteStatementRequest batchRequest =
BatchExecuteStatementRequest.builder()
    .statements(myBatchStatementList)
    .build();

BatchExecuteStatementResponse response =
ddb.batchExecuteStatement(batchRequest);
System.out.println("ExecuteStatement successful: " +
response.toString());
System.out.println("Added new movies using a batch
command.");

} catch (DynamoDbException e) {
```

```
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void updateTableItemBatch(DynamoDbClient ddb) {
    String sqlStatement = "UPDATE MoviesPartiQBatch SET info =
'directors\":[\"Merian C. Cooper\", \"Ernest B. Schoedsack' where year=? and
title=?";
    List<AttributeValue> parametersRec1 = new ArrayList<>();

    // Update three records.
    AttributeValue att1 = AttributeValue.builder()
        .n(String.valueOf("2022"))
        .build();

    AttributeValue att2 = AttributeValue.builder()
        .s("My Movie 1")
        .build();

    parametersRec1.add(att1);
    parametersRec1.add(att2);

    BatchStatementRequest statementRequestRec1 =
BatchStatementRequest.builder()
        .statement(sqlStatement)
        .parameters(parametersRec1)
        .build();

    // Update record 2.
    List<AttributeValue> parametersRec2 = new ArrayList<>();
    AttributeValue attRec2 = AttributeValue.builder()
        .n(String.valueOf("2022"))
        .build();

    AttributeValue attRec2a = AttributeValue.builder()
        .s("My Movie 2")
        .build();

    parametersRec2.add(attRec2);
    parametersRec2.add(attRec2a);
    BatchStatementRequest statementRequestRec2 =
BatchStatementRequest.builder()
        .statement(sqlStatement)
```

```
        .parameters(parametersRec2)
        .build();

// Update record 3.
List<AttributeValue> parametersRec3 = new ArrayList<>();
AttributeValue attRec3 = AttributeValue.builder()
    .n(String.valueOf("2022"))
    .build();

AttributeValue attRec3a = AttributeValue.builder()
    .s("My Movie 3")
    .build();

parametersRec3.add(attRec3);
parametersRec3.add(attRec3a);
BatchStatementRequest statementRequestRec3 =
BatchStatementRequest.builder()
    .statement(sqlStatement)
    .parameters(parametersRec3)
    .build();

// Add all three movies to the list.
List<BatchStatementRequest> myBatchStatementList = new
ArrayList<>();
myBatchStatementList.add(statementRequestRec1);
myBatchStatementList.add(statementRequestRec2);
myBatchStatementList.add(statementRequestRec3);

BatchExecuteStatementRequest batchRequest =
BatchExecuteStatementRequest.builder()
    .statements(myBatchStatementList)
    .build();

try {
    BatchExecuteStatementResponse response =
ddb.batchExecuteStatement(batchRequest);
    System.out.println("ExecuteStatement successful: " +
response.toString());
    System.out.println("Updated three movies using a batch
command.");
} catch (DynamoDbException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
```

```
    }
    System.out.println("Item was updated!");
}

public static void deleteItemBatch(DynamoDbClient ddb) {
    String sqlStatement = "DELETE FROM MoviesPartiQBatch WHERE year = ?
and title=?";
    List<AttributeValue> parametersRec1 = new ArrayList<>();

    // Specify three records to delete.
    AttributeValue att1 = AttributeValue.builder()
        .n(String.valueOf("2022"))
        .build();

    AttributeValue att2 = AttributeValue.builder()
        .s("My Movie 1")
        .build();

    parametersRec1.add(att1);
    parametersRec1.add(att2);

    BatchStatementRequest statementRequestRec1 =
BatchStatementRequest.builder()
        .statement(sqlStatement)
        .parameters(parametersRec1)
        .build();

    // Specify record 2.
    List<AttributeValue> parametersRec2 = new ArrayList<>();
    AttributeValue attRec2 = AttributeValue.builder()
        .n(String.valueOf("2022"))
        .build();

    AttributeValue attRec2a = AttributeValue.builder()
        .s("My Movie 2")
        .build();

    parametersRec2.add(attRec2);
    parametersRec2.add(attRec2a);
    BatchStatementRequest statementRequestRec2 =
BatchStatementRequest.builder()
        .statement(sqlStatement)
        .parameters(parametersRec2)
        .build();
}
```

```
// Specify record 3.
List<AttributeValue> parametersRec3 = new ArrayList<>();
AttributeValue attRec3 = AttributeValue.builder()
    .n(String.valueOf("2022"))
    .build();

AttributeValue attRec3a = AttributeValue.builder()
    .s("My Movie 3")
    .build();

parametersRec3.add(attRec3);
parametersRec3.add(attRec3a);

BatchStatementRequest statementRequestRec3 =
BatchStatementRequest.builder()
    .statement(sqlStatement)
    .parameters(parametersRec3)
    .build();

// Add all three movies to the list.
List<BatchStatementRequest> myBatchStatementList = new
ArrayList<>();
myBatchStatementList.add(statementRequestRec1);
myBatchStatementList.add(statementRequestRec2);
myBatchStatementList.add(statementRequestRec3);

BatchExecuteStatementRequest batchRequest =
BatchExecuteStatementRequest.builder()
    .statements(myBatchStatementList)
    .build();

try {
    ddb.batchExecuteStatement(batchRequest);
    System.out.println("Deleted three movies using a batch
command.");
} catch (DynamoDbException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
}
```

```
public static void deleteDynamoDBTable(DynamoDbClient ddb, String tableName)
{
    DeleteTableRequest request = DeleteTableRequest.builder()
        .tableName(tableName)
        .build();

    try {
        ddb.deleteTable(request);
    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    System.out.println(tableName + " was successfully deleted!");
}

private static ExecuteStatementResponse
executeStatementRequest(DynamoDbClient ddb, String statement,
    List<AttributeValue> parameters) {
    ExecuteStatementRequest request = ExecuteStatementRequest.builder()
        .statement(statement)
        .parameters(parameters)
        .build();

    return ddb.executeStatement(request);
}
}
```

- Per i dettagli sull'API, [BatchExecuteStatement](#) consulta AWS SDK for Java 2.x API Reference.


## Esecuzione di una query mediante PartiQL

L'esempio di codice seguente mostra come:

- Ricezione di un articolo eseguendo un'istruzione SELECT.
- Aggiunta di un elemento eseguendo un'istruzione INSERT.
- Aggiornamento di un elemento eseguendo un'istruzione UPDATE.
- Eliminazione di un elemento eseguendo un'istruzione DELETE.



## SDK per Java 2.x

 Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public class ScenarioPartiQ {
    public static void main(String[] args) throws IOException {
        final String usage = ""

                Usage:
                <fileName>

                Where:
                fileName - The path to the moviedata.json file that you can
download from the Amazon DynamoDB Developer Guide.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String fileName = args[0];
        String tableName = "MoviesPartiQ";
        Region region = Region.US_EAST_1;
        DynamoDbClient ddb = DynamoDbClient.builder()
            .region(region)
            .build();

        System.out.println(
            "***** Creating an Amazon DynamoDB table named MoviesPartiQ with a
key named year and a sort key named title.");
        createTable(ddb, tableName);

        System.out.println("***** Loading data into the MoviesPartiQ table.");
        loadData(ddb, fileName);

        System.out.println("***** Getting data from the MoviesPartiQ table.");
        getItem(ddb);
    }
}
```

```
System.out.println("***** Putting a record into the MoviesPartiQ table.");
putRecord(ddb);

System.out.println("***** Updating a record.");
updateTableItem(ddb);

System.out.println("***** Querying the movies released in 2013.");
queryTable(ddb);

System.out.println("***** Deleting the Amazon DynamoDB table.");
deleteDynamoDBTable(ddb, tableName);
ddb.close();
}

public static void createTable(DynamoDbClient ddb, String tableName) {
    DynamoDbWaiter dbWaiter = ddb.waiter();
    ArrayList<AttributeDefinition> attributeDefinitions = new ArrayList<>();

    // Define attributes.
    attributeDefinitions.add(AttributeDefinition.builder()
        .attributeName("year")
        .attributeType("N")
        .build());

    attributeDefinitions.add(AttributeDefinition.builder()
        .attributeName("title")
        .attributeType("S")
        .build());

    ArrayList<KeySchemaElement> tableKey = new ArrayList<>();
    KeySchemaElement key = KeySchemaElement.builder()
        .attributeName("year")
        .keyType(KeyType.HASH)
        .build();

    KeySchemaElement key2 = KeySchemaElement.builder()
        .attributeName("title")
        .keyType(KeyType.RANGE) // Sort
        .build();

    // Add KeySchemaElement objects to the list.
    tableKey.add(key);
    tableKey.add(key2);
}
```

```

CreateTableRequest request = CreateTableRequest.builder()
    .keySchema(tableKey)
    .provisionedThroughput(ProvisionedThroughput.builder()
        .readCapacityUnits(new Long(10))
        .writeCapacityUnits(new Long(10))
        .build())
    .attributeDefinitions(attributeDefinitions)
    .tableName(tableName)
    .build();

try {
    CreateTableResponse response = ddb.createTable(request);
    DescribeTableRequest tableRequest = DescribeTableRequest.builder()
        .tableName(tableName)
        .build();

    // Wait until the Amazon DynamoDB table is created.
    WaiterResponse<DescribeTableResponse> waiterResponse =
dbWaiter.waitUntilTableExists(tableRequest);
    waiterResponse.matched().response().ifPresent(System.out::println);
    String newTable = response.tableDescription().tableName();
    System.out.println("The " + newTable + " was successfully created.");

} catch (DynamoDbException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}

}

// Load data into the table.
public static void loadData(DynamoDbClient ddb, String fileName) throws
IOException {

    String sqlStatement = "INSERT INTO MoviesPartiQ VALUE {'year':?,
'title' : ?, 'info' : ?}";
    JsonParser parser = new JsonFactory().createParser(new File(fileName));
    com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
    Iterator<JsonNode> iter = rootNode.iterator();
    ObjectNode currentNode;
    int t = 0;
    List<AttributeValue> parameters = new ArrayList<>();
    while (iter.hasNext()) {

```

```
// Add 200 movies to the table.
if (t == 200)
    break;
currentNode = (ObjectNode) iter.next();

int year = currentNode.path("year").asInt();
String title = currentNode.path("title").asText();
String info = currentNode.path("info").toString();

AttributeValue att1 = AttributeValue.builder()
    .n(String.valueOf(year))
    .build();

AttributeValue att2 = AttributeValue.builder()
    .s(title)
    .build();

AttributeValue att3 = AttributeValue.builder()
    .s(info)
    .build();

parameters.add(att1);
parameters.add(att2);
parameters.add(att3);

// Insert the movie into the Amazon DynamoDB table.
executeStatementRequest(ddb, sqlStatement, parameters);
System.out.println("Added Movie " + title);

parameters.remove(att1);
parameters.remove(att2);
parameters.remove(att3);
t++;
}
}

public static void getItem(DynamoDbClient ddb) {

    String sqlStatement = "SELECT * FROM MoviesPartiQ where year=? and title=?";
    List<AttributeValue> parameters = new ArrayList<>();
    AttributeValue att1 = AttributeValue.builder()
        .n("2012")
        .build();
```

```
        AttributeValue att2 = AttributeValue.builder()
            .s("The Perks of Being a Wallflower")
            .build();

        parameters.add(att1);
        parameters.add(att2);

        try {
            ExecuteStatementResponse response = executeStatementRequest(ddb,
                sqlStatement, parameters);
            System.out.println("ExecuteStatement successful: " +
                response.toString());
        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void putRecord(DynamoDbClient ddb) {

        String sqlStatement = "INSERT INTO MoviesPartiQ VALUE {'year':?,
'title' : ?, 'info' : ?}";
        try {
            List<AttributeValue> parameters = new ArrayList<>();

            AttributeValue att1 = AttributeValue.builder()
                .n(String.valueOf("2020"))
                .build();

            AttributeValue att2 = AttributeValue.builder()
                .s("My Movie")
                .build();

            AttributeValue att3 = AttributeValue.builder()
                .s("No Information")
                .build();

            parameters.add(att1);
            parameters.add(att2);
            parameters.add(att3);

            executeStatementRequest(ddb, sqlStatement, parameters);
        }
    }
}
```

```

        System.out.println("Added new movie.");

    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void updateTableItem(DynamoDbClient ddb) {

    String sqlStatement = "UPDATE MoviesPartiQ SET info = 'directors\":[\"Merian
C. Cooper\", \"Ernest B. Schoedsack' where year=? and title=?";
    List<AttributeValue> parameters = new ArrayList<>();
    AttributeValue att1 = AttributeValue.builder()
        .n(String.valueOf("2013"))
        .build();

    AttributeValue att2 = AttributeValue.builder()
        .s("The East")
        .build();

    parameters.add(att1);
    parameters.add(att2);

    try {
        executeStatementRequest(ddb, sqlStatement, parameters);

    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    System.out.println("Item was updated!");
}

// Query the table where the year is 2013.
public static void queryTable(DynamoDbClient ddb) {
    String sqlStatement = "SELECT * FROM MoviesPartiQ where year = ? ORDER BY
year";
    try {

        List<AttributeValue> parameters = new ArrayList<>();
        AttributeValue att1 = AttributeValue.builder()
            .n(String.valueOf("2013"))
            .build();

```

```
        parameters.add(att1);

        // Get items in the table and write out the ID value.
        ExecuteStatementResponse response = executeStatementRequest(ddb,
sqlStatement, parameters);
        System.out.println("ExecuteStatement successful: " +
response.toString());

    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void deleteDynamoDBTable(DynamoDbClient ddb, String tableName) {

    DeleteTableRequest request = DeleteTableRequest.builder()
        .tableName(tableName)
        .build();

    try {
        ddb.deleteTable(request);

    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }

    System.out.println(tableName + " was successfully deleted!");
}

private static ExecuteStatementResponse executeStatementRequest(DynamoDbClient
ddb, String statement,
    List<AttributeValue> parameters) {
    ExecuteStatementRequest request = ExecuteStatementRequest.builder()
        .statement(statement)
        .parameters(parameters)
        .build();

    return ddb.executeStatement(request);
}

private static void processResults(ExecuteStatementResponse
executeStatementResult) {
```

```
        System.out.println("ExecuteStatement successful: " +
            executeStatementResult.toString());
    }
}
```

- Per i dettagli sull'API, [ExecuteStatement](#) consulta AWS SDK for Java 2.x API Reference.

## Interrogazione per elementi TTL

Il seguente esempio di codice mostra come eseguire una query per gli elementi TTL.

### SDK per Java 2.x

Interroga l'espressione filtrata per raccogliere elementi TTL in una tabella DynamoDB.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.QueryRequest;
import software.amazon.awssdk.services.dynamodb.model.QueryResponse;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;
import software.amazon.awssdk.utils.ImmutableMap;

import java.util.Map;
import java.util.Optional;

// Get current time in epoch second format (comparing against expiry
attribute)
final long currentTime = System.currentTimeMillis() / 1000;

// A string that contains conditions that DynamoDB applies after the Query
operation, but before the data is returned to you.
final String keyConditionExpression = "#pk = :pk";

// The condition that specifies the key values for items to be retrieved by
the Query action.
final String filterExpression = "#ea > :ea";
final Map<String, String> expressionAttributeNames = ImmutableMap.of(
    "#pk", "primaryKey",
    "#ea", "expireAt");
```



```

    final Map<String, AttributeValue> expressionAttributeValues =
ImmutableMap.of(
        ":pk", AttributeValue.builder().s(primaryKey).build(),
        ":ea",
AttributeValue.builder().s(String.valueOf(currentTime)).build()
    );

    final QueryRequest request = QueryRequest.builder()
        .tableName(tableName)
        .keyConditionExpression(keyConditionExpression)
        .filterExpression(filterExpression)
        .expressionAttributeNames(expressionAttributeNames)
        .expressionAttributeValues(expressionAttributeValues)
        .build();
    try (DynamoDbClient ddb = DynamoDbClient.builder()
        .region(region)
        .build()) {
        final QueryResponse response = ddb.query(request);
        System.out.println(tableName + " Query operation with TTL successful.
Request id is "
            + response.responseMetadata().requestId());
        // Print the items that are not expired
        for (Map<String, AttributeValue> item : response.items()) {
            System.out.println(item.toString());
        }
    } catch (ResourceNotFoundException e) {
        System.err.format("Error: The Amazon DynamoDB table \"%s\" can't be
found.\n", tableName);
        System.exit(1);
    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    System.exit(0);

```

- Per ulteriori informazioni sulle API, consulta [Query](#) nella Documentazione di riferimento delle API AWS SDK for Java 2.x .

## Aggiorna il TTL di un elemento

Il seguente esempio di codice mostra come aggiornare il TTL di un elemento.

## SDK per Java 2.x

Aggiorna il TTL su un elemento DynamoDB esistente in una tabella.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemRequest;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemResponse;
import software.amazon.awssdk.utils.ImmutableMap;

import java.util.Map;
import java.util.Optional;

// Get current time in epoch second format
final long currentTime = System.currentTimeMillis() / 1000;
// Calculate expiration time 90 days from now in epoch second format
final long expireDate = currentTime + (90 * 24 * 60 * 60);
// An expression that defines one or more attributes to be updated, the
action to be performed on them, and new values for them.
final String updateExpression = "SET updatedAt=:c, expireAt=:e";

final ImmutableMap<String, AttributeValue> keyMap =
    ImmutableMap.of("primaryKey", AttributeValue.fromS(primaryKey),
        "sortKey", AttributeValue.fromS(sortKey));
final Map<String, AttributeValue> expressionAttributeValues =
ImmutableMap.of(
    ":c",
AttributeValue.builder().s(String.valueOf(currentTime)).build(),
    ":e", AttributeValue.builder().s(String.valueOf(expireDate)).build()
);

final UpdateItemRequest request = UpdateItemRequest.builder()
    .tableName(tableName)
    .key(keyMap)
    .updateExpression(updateExpression)
    .expressionAttributeValues(expressionAttributeValues)
    .build();
try (DynamoDbClient ddb = DynamoDbClient.builder()
    .region(region)
    .build()) {
    final UpdateItemResponse response = ddb.updateItem(request);
```

```

        System.out.println(tableName + " UpdateItem operation with TTL
successful. Request id is "
            + response.responseMetadata().requestId());
    } catch (ResourceNotFoundException e) {
        System.err.format("Error: The Amazon DynamoDB table \"%s\" can't be
found.\n", tableName);
        System.exit(1);
    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    System.exit(0);

```

- Per i dettagli sull'API, consulta la sezione API [UpdateItem](#) Reference AWS SDK for Java 2.x .

## Esempi serverless

Segnalazione degli errori degli elementi batch per le funzioni Lambda con un trigger DynamoDB

Il seguente esempio di codice mostra come implementare una risposta batch parziale per le funzioni Lambda che ricevono eventi da un flusso DynamoDB. La funzione riporta gli errori degli elementi batch nella risposta, segnalando a Lambda di riprovare tali messaggi in un secondo momento.

SDK per Java 2.x

### Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri come eseguire la configurazione e l'esecuzione nel repository di [Esempi serverless](#).

Segnalazione degli errori degli elementi batch di DynamoDB con Lambda utilizzando Java.

```

// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.DynamodbEvent;
import com.amazonaws.services.lambda.runtime.events.StreamsEventResponse;
import com.amazonaws.services.lambda.runtime.events.models.dynamodb.StreamRecord;

```

```
import java.io.Serializable;
import java.util.ArrayList;
import java.util.List;

public class ProcessDynamodbRecords implements RequestHandler<DynamodbEvent,
    Serializable> {

    @Override
    public StreamsEventResponse handleRequest(DynamodbEvent input, Context context)
    {

        List<StreamsEventResponse.BatchItemFailure> batchItemFailures = new
        ArrayList<>();
        String curRecordSequenceNumber = "";

        for (DynamodbEvent.DynamodbStreamRecord dynamodbStreamRecord :
        input.getRecords()) {
            try {
                //Process your record
                StreamRecord dynamodbRecord = dynamodbStreamRecord.getDynamodb();
                curRecordSequenceNumber = dynamodbRecord.getSequenceNumber();

            } catch (Exception e) {
                /* Since we are working with streams, we can return the failed item
                immediately.
                Lambda will immediately begin to retry processing from this
                failed item onwards. */
                batchItemFailures.add(new
                StreamsEventResponse.BatchItemFailure(curRecordSequenceNumber));
                return new StreamsEventResponse(batchItemFailures);
            }
        }

        return new StreamsEventResponse();
    }
}
```

## Esempi di Amazon EC2 che utilizzano SDK for Java 2.x

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando Amazon EC2. AWS SDK for Java 2.x

Le operazioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le operazioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati e negli esempi tra servizi.

Scenari: esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio.


Ogni esempio include un collegamento a GitHub, dove puoi trovare istruzioni su come configurare ed eseguire il codice nel contesto.

Nozioni di base

Hello Amazon EC2

Gli esempi di codice seguenti mostrano come iniziare a utilizzare Amazon EC2.

SDK per Java 2.x

 Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public static void describeSecurityGroups(Ec2Client ec2, String groupId) {
    try {
        DescribeSecurityGroupsRequest request =
DescribeSecurityGroupsRequest.builder()
            .groupIds(groupId)
            .build();

        // Use a paginator.
        DescribeSecurityGroupsIterable listGroups =
ec2.describeSecurityGroupsPaginator(request);
        listGroups.stream()
            .flatMap(r -> r.securityGroups().stream())
            .forEach(group -> System.out
                .println(" Group id: " +group.groupId() + " group name = " +
group.groupName()));
    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }
}
```

```
        System.exit(1);
    }
}
```

- Per i dettagli sull'API, [DescribeSecurityGroups](#) consulta AWS SDK for Java 2.x API Reference.

## Argomenti

- [Azioni](#)
- [Scenari](#)

## Azioni

### AllocateAddress

Il seguente esempio di codice mostra come utilizzare `AllocateAddress`.

SDK per Java 2.x

#### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public static String allocateAddress(Ec2Client ec2) {
    try {
        AllocateAddressRequest allocateRequest =
        AllocateAddressRequest.builder()
            .domain(DomainType.VPC)
            .build();

        AllocateAddressResponse allocateResponse =
        ec2.allocateAddress(allocateRequest);
        return allocateResponse.allocationId();

    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

```
    }  
    return "";  
}
```

- Per i dettagli sull'API, [AllocateAddress](#) consulta AWS SDK for Java 2.x API Reference.

## AssociateAddress

Il seguente esempio di codice mostra come utilizzare `AssociateAddress`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public static String associateAddress(Ec2Client ec2, String instanceId, String  
allocationId) {  
    try {  
        AssociateAddressRequest associateRequest =  
AssociateAddressRequest.builder()  
            .instanceId(instanceId)  
            .allocationId(allocationId)  
            .build();  
  
        AssociateAddressResponse associateResponse =  
ec2.associateAddress(associateRequest);  
        return associateResponse.associationId();  
  
    } catch (Ec2Exception e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
    return "";  
}
```

- Per i dettagli sull'API, [AssociateAddress](#) consulta AWS SDK for Java 2.x API Reference.

## AuthorizeSecurityGroupIngress

Il seguente esempio di codice mostra come utilizzare `AuthorizeSecurityGroupIngress`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public static String createSecurityGroup(Ec2Client ec2, String groupName, String
groupDesc, String vpcId,
    String myIpAddress) {
    try {
        CreateSecurityGroupRequest createRequest =
CreateSecurityGroupRequest.builder()
            .groupName(groupName)
            .description(groupDesc)
            .vpcId(vpcId)
            .build();

        CreateSecurityGroupResponse resp =
ec2.createSecurityGroup(createRequest);
        IpRange ipRange = IpRange.builder()
            .cidrIp(myIpAddress + "/0")
            .build();

        IpPermission ipPerm = IpPermission.builder()
            .ipProtocol("tcp")
            .toPort(80)
            .fromPort(80)
            .ipRanges(ipRange)
            .build();

        IpPermission ipPerm2 = IpPermission.builder()
            .ipProtocol("tcp")
            .toPort(22)
            .fromPort(22)
            .ipRanges(ipRange)
            .build();
```



```
        AuthorizeSecurityGroupIngressRequest authRequest =
AuthorizeSecurityGroupIngressRequest.builder()
    .groupName(groupName)
    .ipPermissions(ipPerm, ipPerm2)
    .build();

    ec2.authorizeSecurityGroupIngress(authRequest);
    System.out.println("Successfully added ingress policy to security group
" + groupName);
    return resp.groupId();

    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
```

- Per i dettagli sull'API, [AuthorizeSecurityGroupIngress](#) consulta AWS SDK for Java 2.x API Reference.

## CreateKeyPair

Il seguente esempio di codice mostra come utilizzare `CreateKeyPair`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public static void createKeyPair(Ec2Client ec2, String keyName, String fileName)
{
    try {
        CreateKeyPairRequest request = CreateKeyPairRequest.builder()
            .keyName(keyName)
            .build();
```

```
        CreateKeyPairResponse response = ec2.createKeyPair(request);
        String content = response.keyMaterial();
        BufferedWriter writer = new BufferedWriter(new FileWriter(fileName));
        writer.write(content);
        writer.close();
        System.out.println("Successfully created key pair named " + keyName);

    } catch (Ec2Exception | IOException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- Per i dettagli sull'API, [CreateKeyPair](#) consulta AWS SDK for Java 2.x API Reference.

## CreateSecurityGroup

Il seguente esempio di codice mostra come utilizzare `CreateSecurityGroup`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public static String createSecurityGroup(Ec2Client ec2, String groupName, String
groupDesc, String vpcId,
    String myIpAddress) {
    try {
        CreateSecurityGroupRequest createRequest =
CreateSecurityGroupRequest.builder()
            .groupName(groupName)
            .description(groupDesc)
            .vpcId(vpcId)
            .build();

        CreateSecurityGroupResponse resp =
ec2.createSecurityGroup(createRequest);
```

```
        IpRange ipRange = IpRange.builder()
            .cidrIp(myIpAddress + "/0")
            .build();

        IpPermission ipPerm = IpPermission.builder()
            .ipProtocol("tcp")
            .toPort(80)
            .fromPort(80)
            .ipRanges(ipRange)
            .build();

        IpPermission ipPerm2 = IpPermission.builder()
            .ipProtocol("tcp")
            .toPort(22)
            .fromPort(22)
            .ipRanges(ipRange)
            .build();

        AuthorizeSecurityGroupIngressRequest authRequest =
AuthorizeSecurityGroupIngressRequest.builder()
            .groupName(groupName)
            .ipPermissions(ipPerm, ipPerm2)
            .build();

        ec2.authorizeSecurityGroupIngress(authRequest);
        System.out.println("Successfully added ingress policy to security group
" + groupName);
        return resp.groupId();


    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
```

- Per i dettagli sull'API, [CreateSecurityGroup](#) consulta AWS SDK for Java 2.x API Reference.

## DeleteKeyPair

Il seguente esempio di codice mostra come utilizzare `DeleteKeyPair`.

## SDK per Java 2.x

 Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public static void deleteKeys(Ec2Client ec2, String keyPair) {
    try {
        DeleteKeyPairRequest request = DeleteKeyPairRequest.builder()
            .keyName(keyPair)
            .build();


        ec2.deleteKeyPair(request);
        System.out.println("Successfully deleted key pair named " + keyPair);
    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Per i dettagli sull'API, [DeleteKeyPair](#) consulta AWS SDK for Java 2.x API Reference.

## DeleteSecurityGroup

Il seguente esempio di codice mostra come utilizzare `DeleteSecurityGroup`.

## SDK per Java 2.x

 Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public static void deleteEC2SecGroup(Ec2Client ec2, String groupId) {
    try {
```

```
        DeleteSecurityGroupRequest request =
DeleteSecurityGroupRequest.builder()
        .groupId(groupId)
        .build();

        ec2.deleteSecurityGroup(request);
        System.out.println("Successfully deleted security group with Id " +
groupId);

    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Per i dettagli sull'API, [DeleteSecurityGroup](#) consulta AWS SDK for Java 2.x API Reference.

## DescribeInstanceTypes

Il seguente esempio di codice mostra come utilizzare `DescribeInstanceTypes`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
// Get a list of instance types.
public static String getInstanceTypes(Ec2Client ec2) {
    String instanceType;
    try {
        DescribeInstanceTypesRequest typesRequest =
DescribeInstanceTypesRequest.builder()
        .maxResults(10)
        .build();

        DescribeInstanceTypesResponse response =
ec2.describeInstanceTypes(typesRequest);
        List<InstanceTypeInfo> instanceTypes = response.instanceTypes();
```

```

        for (InstanceTypeInfo type : instanceTypes) {
            System.out.println("The memory information of this type is " +
                type.memoryInfo().sizeInMiB());
            System.out.println("Network information is " +
                type.networkInfo().toString());
            System.out.println("Instance type is " +
                type.instanceType().toString());
            instanceType = type.instanceType().toString();
            if (instanceType.compareTo("t2.2xlarge") == 0){
                return instanceType;
            }
        }

    } catch (SsmException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return "";
}

```

- Per i dettagli sull'API, [DescribeInstanceTypes](#) consulta AWS SDK for Java 2.x API Reference.

## DescribeInstances

Il seguente esempio di codice mostra come utilizzare `DescribeInstances`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.DescribeInstancesRequest;
import software.amazon.awssdk.services.ec2.model.Ec2Exception;
import software.amazon.awssdk.services.ec2.paginators.DescribeInstancesIterable;

/**

```

```
* Before running this Java V2 code example, set up your development
* environment, including your credentials.
*
* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class DescribeInstances {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        Ec2Client ec2 = Ec2Client.builder()
            .region(region)
            .build();

        describeEC2Instances(ec2);
        ec2.close();
    }

    public static void describeEC2Instances(Ec2Client ec2) {
        try {
            DescribeInstancesRequest request = DescribeInstancesRequest.builder()
                .maxResults(10)
                .build();

            DescribeInstancesIterable instancesIterable =
ec2.describeInstancesPaginator(request);
            instancesIterable.stream()
                .flatMap(r -> r.reservations().stream())
                .flatMap(reservation -> reservation.instances().stream())
                .forEach(instance -> {
                    System.out.println("Instance Id is " + instance.instanceId());
                    System.out.println("Image id is " + instance.imageId());
                    System.out.println("Instance type is " +
instance.instanceType());
                    System.out.println("Instance state name is " +
instance.state().name());
                    System.out.println("Monitoring information is " +
instance.monitoring().state());
                });
        } catch (Ec2Exception e) {
            System.err.println(e.awsErrorDetails().errorCode());
            System.exit(1);
        }
    }
}
```

```
}  
}
```

- Per i dettagli sull'API, [DescribeInstances](#) consulta AWS SDK for Java 2.x API Reference.

## DescribeKeyPairs

Il seguente esempio di codice mostra come utilizzare `DescribeKeyPairs`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public static void describeKeys(Ec2Client ec2) {  
    try {  
        DescribeKeyPairsResponse response = ec2.describeKeyPairs();  
        response.keyPairs().forEach(keyPair -> System.out.printf(  
            "Found key pair with name %s " +  
            "and fingerprint %s",  
            keyPair.keyName(),  
            keyPair.keyFingerprint()));  
    } catch (Ec2Exception e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

- Per i dettagli sull'API, [DescribeKeyPairs](#) consulta AWS SDK for Java 2.x API Reference.

## DescribeSecurityGroups

Il seguente esempio di codice mostra come utilizzare `DescribeSecurityGroups`.



## SDK per Java 2.x

 Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public static void describeSecurityGroups(Ec2Client ec2, String groupId) {
    try {
        DescribeSecurityGroupsRequest request =
DescribeSecurityGroupsRequest.builder()
            .groupIds(groupId)
            .build();


        // Use a paginator.
        DescribeSecurityGroupsIterable listGroups =
ec2.describeSecurityGroupsPaginator(request);
        listGroups.stream()
            .flatMap(r -> r.securityGroups().stream())
            .forEach(group -> System.out
                .println(" Group id: " +group.groupId() + " group name = " +
group.groupName()));
    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Per i dettagli sull'API, [DescribeSecurityGroups](#) consulta AWS SDK for Java 2.x API Reference.

## DisassociateAddress

Il seguente esempio di codice mostra come utilizzare `DisassociateAddress`.

## SDK per Java 2.x

 Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public static void disassociateAddress(Ec2Client ec2, String associationId) {
    try {
        DisassociateAddressRequest addressRequest =
DisassociateAddressRequest.builder()
            .associationId(associationId)
            .build();


        ec2.disassociateAddress(addressRequest);
        System.out.println("You successfully disassociated the address!");
    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Per i dettagli sull'API, [DisassociateAddress](#) consulta AWS SDK for Java 2.x API Reference.

## ReleaseAddress

Il seguente esempio di codice mostra come utilizzare `ReleaseAddress`.

## SDK per Java 2.x

 Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public static void releaseEC2Address(Ec2Client ec2, String allocId) {
```

```
    try {
        ReleaseAddressRequest request = ReleaseAddressRequest.builder()
            .allocationId(allocId)
            .build();

        ec2.releaseAddress(request);
        System.out.println("Successfully released Elastic IP address " +
allocId);
    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Per i dettagli sull'API, [ReleaseAddress](#) consulta AWS SDK for Java 2.x API Reference.

## RunInstances

Il seguente esempio di codice mostra come utilizzare `RunInstances`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.InstanceType;
import software.amazon.awssdk.services.ec2.model.RunInstancesRequest;
import software.amazon.awssdk.services.ec2.model.RunInstancesResponse;
import software.amazon.awssdk.services.ec2.model.Tag;
import software.amazon.awssdk.services.ec2.model.CreateTagsRequest;
import software.amazon.awssdk.services.ec2.model.Ec2Exception;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 */
```

```
* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*
* This code example requires an AMI value. You can learn more about this value
* by reading this documentation topic:
*
* https://docs.aws.amazon.com/AWSEC2/latest/WindowsGuide/AMIs.html
*/
public class CreateInstance {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <name> <amiId>

            Where:
                name - An instance name value that you can obtain from the AWS
Console (for example, ami-xxxxxx5c8b987b1a0).\s
                amiId - An Amazon Machine Image (AMI) value that you can obtain
from the AWS Console (for example, i-xxxxxx2734106d0ab).\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String name = args[0];
        String amiId = args[1];
        Region region = Region.US_EAST_1;
        Ec2Client ec2 = Ec2Client.builder()
            .region(region)
            .build();

        String instanceId = createEC2Instance(ec2, name, amiId);
        System.out.println("The Amazon EC2 Instance ID is " + instanceId);
        ec2.close();
    }

    public static String createEC2Instance(Ec2Client ec2, String name, String amiId)
    {
        RunInstancesRequest runRequest = RunInstancesRequest.builder()
            .imageId(amiId)
```

```
        .instanceType(InstanceType.T1_MICRO)
        .maxCount(1)
        .minCount(1)
        .build();

// Use a waiter to wait until the instance is running.
System.out.println("Going to start an EC2 instance using a waiter");
RunInstancesResponse response = ec2.runInstances(runRequest);
String instanceIdVal = response.instances().get(0).instanceId();
ec2.waiter().waitUntilInstanceRunning(r -> r.instanceIds(instanceIdVal));
Tag tag = Tag.builder()
    .key("Name")
    .value(name)
    .build();

CreateTagsRequest tagRequest = CreateTagsRequest.builder()
    .resources(instanceIdVal)
    .tags(tag)
    .build();

try {
    ec2.createTags(tagRequest);
    System.out.printf("Successfully started EC2 Instance %s based on AMI %s", instanceIdVal, amiId);
    return instanceIdVal;
} catch (Ec2Exception e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}

return "";
}
}
```

- Per i dettagli sull'API, [RunInstances](#) consulta AWS SDK for Java 2.x API Reference.

## StartInstances

Il seguente esempio di codice mostra come utilizzare `StartInstances`.

## SDK per Java 2.x

 Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public static void startInstance(Ec2Client ec2, String instanceId) {
    Ec2Waiter ec2Waiter = Ec2Waiter.builder()
        .overrideConfiguration(b -> b.maxAttempts(100))
        .client(ec2)
        .build();

    StartInstancesRequest request = StartInstancesRequest.builder()
        .instanceIds(instanceId)
        .build();

    System.out.println("Use an Ec2Waiter to wait for the instance to run. This
will take a few minutes.");
    ec2.startInstances(request);
    DescribeInstancesRequest instanceRequest =
DescribeInstancesRequest.builder()
        .instanceIds(instanceId)
        .build();

    WaiterResponse<DescribeInstancesResponse> waiterResponse =
ec2Waiter.waitUntilInstanceRunning(instanceRequest);
    waiterResponse.matched().response().ifPresent(System.out::println);
    System.out.println("Successfully started instance " + instanceId);
}
```

- Per i dettagli sull'API, [StartInstances](#) consulta AWS SDK for Java 2.x API Reference.

## StopInstances

Il seguente esempio di codice mostra come utilizzare `StopInstances`.

## SDK per Java 2.x

 Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public static void stopInstance(Ec2Client ec2, String instanceId) {
    Ec2Waiter ec2Waiter = Ec2Waiter.builder()
        .overrideConfiguration(b -> b.maxAttempts(100))
        .client(ec2)
        .build();
    StopInstancesRequest request = StopInstancesRequest.builder()
        .instanceIds(instanceId)
        .build();

    System.out.println("Use an Ec2Waiter to wait for the instance to stop. This
will take a few minutes.");
    ec2.stopInstances(request);
    DescribeInstancesRequest instanceRequest =
DescribeInstancesRequest.builder()
        .instanceIds(instanceId)
        .build();

    WaiterResponse<DescribeInstancesResponse> waiterResponse =
ec2Waiter.waitUntilInstanceStopped(instanceRequest);
    waiterResponse.matched().response().ifPresent(System.out::println);
    System.out.println("Successfully stopped instance " + instanceId);
}
```

- Per i dettagli sull'API, [StopInstances](#) consulta AWS SDK for Java 2.x API Reference.

## TerminateInstances

Il seguente esempio di codice mostra come utilizzare `TerminateInstances`.

## SDK per Java 2.x

 Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public static void terminateEC2(Ec2Client ec2, String instanceId) {
    try {
        Ec2Waiter ec2Waiter = Ec2Waiter.builder()
            .overrideConfiguration(b -> b.maxAttempts(100))
            .client(ec2)
            .build();

        TerminateInstancesRequest ti = TerminateInstancesRequest.builder()
            .instanceIds(instanceId)
            .build();

        System.out.println("Use an Ec2Waiter to wait for the instance to
terminate. This will take a few minutes.");
        ec2.terminateInstances(ti);
        DescribeInstancesRequest instanceRequest =
DescribeInstancesRequest.builder()
            .instanceIds(instanceId)
            .build();

        WaiterResponse<DescribeInstancesResponse> waiterResponse = ec2Waiter
            .waitUntilInstanceTerminated(instanceRequest);
        waiterResponse.matched().response().ifPresent(System.out::println);
        System.out.println("Successfully started instance " + instanceId);
        System.out.println(instanceId + " is terminated!");

    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Per i dettagli sull'API, [TerminateInstances](#) consulta AWS SDK for Java 2.x API Reference.



## Scenari

### Creazione e gestione di un servizio resiliente

Il seguente esempio di codice mostra come creare un servizio Web con bilanciamento del carico che restituisca consigli su libri, film e canzoni. L'esempio mostra come il servizio risponde ai guasti e spiega come ristrutturarlo per una maggiore resilienza in caso di guasti.

- Utilizza un gruppo con dimensionamento automatico Amazon EC2 per creare istanze Amazon Elastic Compute Cloud (Amazon EC2) basate su un modello di avvio e per mantenere il numero di istanze entro un intervallo specificato.
- Gestisci e distribuisce le richieste HTTP con Elastic Load Balancing.
- Monitora lo stato delle istanze in un gruppo con dimensionamento automatico e inoltra le richieste soltanto alle istanze integre.
- Esegui un server Web Python su ogni istanza EC2 per gestire le richieste HTTP. Il server Web risponde con consigli e controlli dell'integrità.
- Simula un servizio di raccomandazione con una tabella Amazon DynamoDB.
- Controlla la risposta del server web alle richieste e ai controlli sanitari aggiornando AWS Systems Manager i parametri.

### SDK per Java 2.x

#### Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Esegui lo scenario interattivo al prompt dei comandi.

```
public class Main {  
  
    public static final String fileName = "C:\\AWS\\resworkflow\\  
\\recommendations.json"; // Modify file location.  
    public static final String tableName = "doc-example-recommendation-service";  
    public static final String startScript = "C:\\AWS\\resworkflow\\  
\\server_startup_script.sh"; // Modify file location.  
}
```

```
    public static final String policyFile = "C:\\\\AWS\\\\resworkflow\\
\\instance_policy.json"; // Modify file location.
    public static final String ssmJSON = "C:\\\\AWS\\\\resworkflow\\
\\ssm_only_policy.json"; // Modify file location.
    public static final String failureResponse = "doc-example-resilient-
architecture-failure-response";
    public static final String healthCheck = "doc-example-resilient-architecture-
health-check";
    public static final String templateName = "doc-example-resilience-template";
    public static final String roleName = "doc-example-resilience-role";
    public static final String policyName = "doc-example-resilience-pol";
    public static final String profileName = "doc-example-resilience-prof";

    public static final String badCredsProfileName = "doc-example-resilience-prof-
bc";

    public static final String targetGroupName = "doc-example-resilience-tg";
    public static final String autoScalingGroupName = "doc-example-resilience-
group";
    public static final String lbName = "doc-example-resilience-lb";
    public static final String protocol = "HTTP";
    public static final int port = 80;

    public static final String DASHES = new String(new char[80]).replace("\\0", "-");

    public static void main(String[] args) throws IOException, InterruptedException
    {
        Scanner in = new Scanner(System.in);
        Database database = new Database();
        AutoScaler autoScaler = new AutoScaler();
        LoadBalancer loadBalancer = new LoadBalancer();

        System.out.println(DASHES);
        System.out.println("Welcome to the demonstration of How to Build and Manage
a Resilient Service!");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("A - SETUP THE RESOURCES");
        System.out.println("Press Enter when you're ready to start deploying
resources.");
        in.nextLine();
        deploy(loadBalancer);
        System.out.println(DASHES);
    }
}
```

```

        System.out.println(DASHES);
        System.out.println("B - DEMO THE RESILIENCE FUNCTIONALITY");
        System.out.println("Press Enter when you're ready.");
        in.nextLine();
        demo(loadBalancer);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("C - DELETE THE RESOURCES");
        System.out.println("""
            This concludes the demo of how to build and manage a resilient
service.

            To keep things tidy and to avoid unwanted charges on your account,
we can clean up all AWS resources
            that were created for this demo.
            """);

        System.out.println("\n Do you want to delete the resources (y/n)? ");
        String userInput = in.nextLine().trim().toLowerCase(); // Capture user input

        if (userInput.equals("y")) {
            // Delete resources here
            deleteResources(loadBalancer, autoScaler, database);
            System.out.println("Resources deleted.");
        } else {
            System.out.println("""
                Okay, we'll leave the resources intact.
                Don't forget to delete them when you're done with them or you
might incur unexpected charges.
            """);
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("The example has completed. ");
        System.out.println("\n Thanks for watching!");
        System.out.println(DASHES);
    }

    // Deletes the AWS resources used in this example.
    private static void deleteResources(LoadBalancer loadBalancer, AutoScaler
autoScaler, Database database)
        throws IOException, InterruptedException {
        loadBalancer.deleteLoadBalancer(lbName);

```

```

        System.out.println("*** Wait 30 secs for resource to be deleted");
        TimeUnit.SECONDS.sleep(30);
        loadBalancer.deleteTargetGroup(targetGroupName);
        autoScaler.deleteAutoScalingGroup(autoScalingGroupName);
        autoScaler.deleteRolesPolicies(policyName, roleName, profileName);
        autoScaler.deleteTemplate(templateName);
        database.deleteTable(tableName);
    }

    private static void deploy(LoadBalancer loadBalancer) throws
    InterruptedException, IOException {
        Scanner in = new Scanner(System.in);
        System.out.println(
            """
                For this demo, we'll use the AWS SDK for Java (v2) to create
    several AWS resources
                to set up a load-balanced web service endpoint and explore
    some ways to make it resilient
                against various kinds of failures.

                Some of the resources create by this demo are:
                \t* A DynamoDB table that the web service depends on to
    provide book, movie, and song recommendations.
                \t* An EC2 launch template that defines EC2 instances that
    each contain a Python web server.
                \t* An EC2 Auto Scaling group that manages EC2 instances
    across several Availability Zones.
                \t* An Elastic Load Balancing (ELB) load balancer that
    targets the Auto Scaling group to distribute requests.
            """);

        System.out.println("Press Enter when you're ready.");
        in.nextLine();
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Creating and populating a DynamoDB table named " +
        tableName);
        Database database = new Database();
        database.createTable(tableName, fileName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("""

```

Creating an EC2 launch template that runs '{startup\_script}' when an instance starts.

This script starts a Python web server defined in the `server.py` script. The web server

listens to HTTP requests on port 80 and responds to requests to '/' and to '/healthcheck'.

For demo purposes, this server is run as the root user. In production, the best practice is to run a web server, such as Apache, with least-privileged credentials.

The template also defines an IAM policy that each instance uses to assume a role that grants permissions to access the DynamoDB recommendation table and Systems Manager parameters that control the flow of the demo.

```
""");
```

```
LaunchTemplateCreator templateCreator = new LaunchTemplateCreator();
templateCreator.createTemplate(policyFile, policyName, profileName,
startScript, templateName, roleName);
System.out.println(DASHES);
```

```
System.out.println(DASHES);
System.out.println(
    "Creating an EC2 Auto Scaling group that maintains three EC2
instances, each in a different Availability Zone.");
System.out.println("*** Wait 30 secs for the VPC to be created");
TimeUnit.SECONDS.sleep(30);
AutoScaler autoScaler = new AutoScaler();
String[] zones = autoScaler.createGroup(3, templateName,
autoScalingGroupName);
```

```
System.out.println("""
    At this point, you have EC2 instances created. Once each instance
starts, it listens for
    HTTP requests. You can see these instances in the console or
continue with the demo.
    Press Enter when you're ready to continue.
""");
```

```
in.nextLine();
System.out.println(DASHES);
```

```
System.out.println(DASHES);
```

```
System.out.println("Creating variables that control the flow of the demo.");
ParameterHelper paramHelper = new ParameterHelper();
paramHelper.reset();
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("""
    Creating an Elastic Load Balancing target group and load balancer.
The target group
    defines how the load balancer connects to instances. The load
balancer provides a
    single endpoint where clients connect and dispatches requests to
instances in the group.
    """);

String vpcId = autoScaler.getDefaultVPC();
List<Subnet> subnets = autoScaler.getSubnets(vpcId, zones);
System.out.println("You have retrieved a list with " + subnets.size() + "
subnets");
String targetGroupArn = loadBalancer.createTargetGroup(protocol, port,
vpcId, targetGroupName);
String elbDnsName = loadBalancer.createLoadBalancer(subnets, targetGroupArn,
lbName, port, protocol);
autoScaler.attachLoadBalancerTargetGroup(autoScalingGroupName,
targetGroupArn);
System.out.println("Verifying access to the load balancer endpoint...");
boolean wasSuccessful = loadBalancer.verifyLoadBalancerEndpoint(elbDnsName);
if (!wasSuccessful) {
    System.out.println("Couldn't connect to the load balancer, verifying
that the port is open...");
    CloseableHttpClient httpClient = HttpClients.createDefault();

    // Create an HTTP GET request to "http://checkip.amazonaws.com"
    HttpGet httpGet = new HttpGet("http://checkip.amazonaws.com");
    try {
        // Execute the request and get the response
        HttpResponse response = httpClient.execute(httpGet);

        // Read the response content.
        String ipAddress =
IOUtils.toString(response.getEntity().getContent(), StandardCharsets.UTF_8).trim();

        // Print the public IP address.
        System.out.println("Public IP Address: " + ipAddress);
```

```
        GroupInfo groupInfo = autoScaler.verifyInboundPort(vpcId, port,
ipAddress);
        if (!groupInfo.isPortOpen()) {
            System.out.println("""
                For this example to work, the default security group for
your default VPC must
                allow access from this computer. You can either add it
automatically from this
                example or add it yourself using the AWS Management
Console.
                """);

            System.out.println(
                "Do you want to add a rule to security group " +
groupInfo.getGroupName() + " to allow");
            System.out.println("inbound traffic on port " + port + " from
your computer's IP address (y/n) ");
            String ans = in.nextLine();
            if ("y".equalsIgnoreCase(ans)) {
                autoScaler.openInboundPort(groupInfo.getGroupName(),
String.valueOf(port), ipAddress);
                System.out.println("Security group rule added.");
            } else {
                System.out.println("No security group rule added.");
            }
        }

    } catch (AutoScalingException e) {
        e.printStackTrace();
    }
} else if (wasSuccessful) {
    System.out.println("Your load balancer is ready. You can access it by
browsing to:");
    System.out.println("\t http://" + elbDnsName);
} else {
    System.out.println("Couldn't get a successful response from the load
balancer endpoint. Troubleshoot by");
    System.out.println("manually verifying that your VPC and security group
are configured correctly and that");
    System.out.println("you can successfully make a GET request to the load
balancer.");
}
```

```

        System.out.println("Press Enter when you're ready to continue with the
demo.");
        in.nextLine();
    }

    // A method that controls the demo part of the Java program.
    public static void demo(LoadBalancer loadBalancer) throws IOException,
InterruptedException {
        ParameterHelper paramHelper = new ParameterHelper();
        System.out.println("Read the ssm_only_policy.json file");
        String ssmOnlyPolicy = readFileAsString(ssmJSON);

        System.out.println("Resetting parameters to starting values for demo.");
        paramHelper.reset();

        System.out.println(
            """
                This part of the demonstration shows how to toggle
different parts of the system
                to create situations where the web service fails, and shows
how using a resilient
                architecture can keep the web service running in spite of
these failures.

                At the start, the load balancer endpoint returns
recommendations and reports that all targets are healthy.
            """);
        demoChoices(loadBalancer);

        System.out.println(
            """
                The web service running on the EC2 instances gets
recommendations by querying a DynamoDB table.
                The table name is contained in a Systems Manager parameter
named self.param_helper.table.
                To simulate a failure of the recommendation service, let's
set this parameter to name a non-existent table.
            """);
        paramHelper.put(paramHelper.tableName, "this-is-not-a-table");

        System.out.println(
            """
                \nNow, sending a GET request to the load balancer endpoint
returns a failure code. But, the service reports as

```



```

        healthy to the load balancer because shallow health checks
don't check for failure of the recommendation service.
        """);
demoChoices(loadBalancer);

System.out.println(
    ""
        Instead of failing when the recommendation service fails,
the web service can return a static response.
        While this is not a perfect solution, it presents the
customer with a somewhat better experience than failure.
        """);
paramHelper.put(paramHelper.failureResponse, "static");

System.out.println("""
    Now, sending a GET request to the load balancer endpoint returns a
static response.
    The service still reports as healthy because health checks are still
shallow.
        """);
demoChoices(loadBalancer);

System.out.println("Let's reinstate the recommendation service.");
paramHelper.put(paramHelper.tableName, paramHelper.dyntable);

System.out.println("""
    Let's also substitute bad credentials for one of the instances in
the target group so that it can't
    access the DynamoDB recommendation table. We will get an instance id
value.
        """);

LaunchTemplateCreator templateCreator = new LaunchTemplateCreator();
AutoScaler autoScaler = new AutoScaler();

// Create a new instance profile based on badCredsProfileName.
templateCreator.createInstanceProfile(policyFile, policyName,
badCredsProfileName, roleName);
String badInstanceId = autoScaler.getBadInstance(autoScalingGroupName);
System.out.println("The bad instance id values used for this demo is " +
badInstanceId);

String profileAssociationId = autoScaler.getInstanceProfile(badInstanceId);
System.out.println("The association Id value is " + profileAssociationId);

```

```
System.out.println("Replacing the profile for instance " + badInstanceId
    + " with a profile that contains bad credentials");
autoScaler.replaceInstanceProfile(badInstanceId, badCredsProfileName,
profileAssociationId);

System.out.println(
    ""
        Now, sending a GET request to the load balancer endpoint
returns either a recommendation or a static response,
        depending on which instance is selected by the load
balancer.
    "");

demoChoices(loadBalancer);

System.out.println("""
    Let's implement a deep health check. For this demo, a deep health
check tests whether
    the web service can access the DynamoDB table that it depends on for
recommendations. Note that
    the deep health check is only for ELB routing and not for Auto
Scaling instance health.
    This kind of deep health check is not recommended for Auto Scaling
instance health, because it
    risks accidental termination of all instances in the Auto Scaling
group when a dependent service fails.
    """);

System.out.println("""
    By implementing deep health checks, the load balancer can detect
when one of the instances is failing
    and take that instance out of rotation.
    """);

paramHelper.put(paramHelper.healthCheck, "deep");

System.out.println("""
    Now, checking target health indicates that the instance with bad
credentials
    is unhealthy. Note that it might take a minute or two for the load
balancer to detect the unhealthy
    instance. Sending a GET request to the load balancer endpoint always
returns a recommendation, because
    the load balancer takes unhealthy instances out of its rotation.
```

```
        """);

        demoChoices(loadBalancer);

        System.out.println(
            ""
                "Because the instances in this demo are controlled by an auto
scaler, the simplest way to fix an unhealthy
                instance is to terminate it and let the auto scaler start a
new instance to replace it.
                """);
        autoScaler.terminateInstance(badInstanceId);

        System.out.println("""
            Even while the instance is terminating and the new instance is
starting, sending a GET
            request to the web service continues to get a successful
recommendation response because
            the load balancer routes requests to the healthy instances. After
the replacement instance
            starts and reports as healthy, it is included in the load balancing
rotation.

            Note that terminating and replacing an instance typically takes
several minutes, during which time you
            can see the changing health check status until the new instance is
running and healthy.
            """);

        demoChoices(loadBalancer);
        System.out.println(
            "If the recommendation service fails now, deep health checks mean
all instances report as unhealthy.");
        paramHelper.put(paramHelper.tableName, "this-is-not-a-table");

        demoChoices(loadBalancer);
        paramHelper.reset();
    }

    public static void demoChoices(LoadBalancer loadBalancer) throws IOException,
InterruptedException {
        String[] actions = {
            "Send a GET request to the load balancer endpoint.",
            "Check the health of load balancer targets.",
            "Go to the next part of the demo."
        }
    }
}
```

```
};
Scanner scanner = new Scanner(System.in);

while (true) {
    System.out.println("-".repeat(88));
    System.out.println("See the current state of the service by selecting
one of the following choices:");
    for (int i = 0; i < actions.length; i++) {
        System.out.println(i + ": " + actions[i]);
    }

    try {
        System.out.print("\nWhich action would you like to take? ");
        int choice = scanner.nextInt();
        System.out.println("-".repeat(88));

        switch (choice) {
            case 0 -> {
                System.out.println("Request:\n");
                System.out.println("GET http://" +
loadBalancer.getEndpoint(lbName));
                CloseableHttpClient httpClient =
HttpClients.createDefault();

                // Create an HTTP GET request to the ELB.
                HttpGet httpGet = new HttpGet("http://" +
loadBalancer.getEndpoint(lbName));

                // Execute the request and get the response.
                HttpResponse response = httpClient.execute(httpGet);
                int statusCode = response.getStatusLine().getStatusCode();
                System.out.println("HTTP Status Code: " + statusCode);

                // Display the JSON response
                BufferedReader reader = new BufferedReader(
                    new
InputStreamReader(response.getEntity().getContent()));
                StringBuilder jsonResponse = new StringBuilder();
                String line;
                while ((line = reader.readLine()) != null) {
                    jsonResponse.append(line);
                }
                reader.close();
            }
        }
    }
}
```

```

        // Print the formatted JSON response.
        System.out.println("Full Response:\n");
        System.out.println(jsonResponse.toString());

        // Close the HTTP client.
        httpClient.close();

    }
    case 1 -> {
        System.out.println("\nChecking the health of load balancer
targets:\n");

        List<TargetHealthDescription> health =
loadBalancer.checkTargetHealth(targetGroupName);
        for (TargetHealthDescription target : health) {
            System.out.printf("\tTarget %s on port %d is %s\n",
target.target().id(),
                                target.target().port(),
target.targetHealth().stateAsString());
        }
        System.out.println("""
Note that it can take a minute or two for the health
check to update
                                after changes are made.
                                """);
    }
    case 2 -> {
        System.out.println("\nOkay, let's move on.");
        System.out.println("-".repeat(88));
        return; // Exit the method when choice is 2
    }
    default -> System.out.println("You must choose a value between
0-2. Please select again.");
}

} catch (java.util.InputMismatchException e) {
    System.out.println("Invalid input. Please select again.");
    scanner.nextLine(); // Clear the input buffer.
}
}
}

public static String readFileAsString(String filePath) throws IOException {
    byte[] bytes = Files.readAllBytes(Paths.get(filePath));
    return new String(bytes);
}

```

```
}  
}
```

Crea una classe che racchiuda le operazioni di dimensionamento automatico e Amazon EC2.

```
public class AutoScaler {  
  
    private static Ec2Client ec2Client;  
    private static AutoScalingClient autoScalingClient;  
    private static IamClient iamClient;  
  
    private static SsmClient ssmClient;  
  
    private IamClient getIAMClient() {  
        if (iamClient == null) {  
            iamClient = IamClient.builder()  
                .region(Region.US_EAST_1)  
                .build();  
        }  
        return iamClient;  
    }  
  
    private SsmClient getSSMClient() {  
        if (ssmClient == null) {  
            ssmClient = SsmClient.builder()  
                .region(Region.US_EAST_1)  
                .build();  
        }  
        return ssmClient;  
    }  
  
    private Ec2Client getEc2Client() {  
        if (ec2Client == null) {  
            ec2Client = Ec2Client.builder()  
                .region(Region.US_EAST_1)  
                .build();  
        }  
        return ec2Client;  
    }  
  
    private AutoScalingClient getAutoScalingClient() {  
        if (autoScalingClient == null) {
```

```

        autoScalingClient = AutoScalingClient.builder()
            .region(Region.US_EAST_1)
            .build();
    }
    return autoScalingClient;
}

/**
 * Terminates and instances in an EC2 Auto Scaling group. After an instance is
 * terminated, it can no longer be accessed.
 */
public void terminateInstance(String instanceId) {
    TerminateInstanceInAutoScalingGroupRequest terminateInstanceIRequest =
    TerminateInstanceInAutoScalingGroupRequest
        .builder()
        .instanceId(instanceId)
        .shouldDecrementDesiredCapacity(false)
        .build();

    getAutoScalingClient().terminateInstanceInAutoScalingGroup(terminateInstanceIRequest);
    System.out.format("Terminated instance %s.", instanceId);
}

/**
 * Replaces the profile associated with a running instance. After the profile is
 * replaced, the instance is rebooted to ensure that it uses the new profile.
 * When
 * the instance is ready, Systems Manager is used to restart the Python web
 * server.
 */
public void replaceInstanceProfile(String instanceId, String
newInstanceProfileName, String profileAssociationId)
    throws InterruptedException {
    // Create an IAM instance profile specification.
    software.amazon.awssdk.services.ec2.model.IamInstanceProfileSpecification
iamInstanceProfile =
software.amazon.awssdk.services.ec2.model.IamInstanceProfileSpecification
        .builder()
        .name(newInstanceProfileName) // Make sure 'newInstanceProfileName'
is a valid IAM Instance Profile
                                // name.
        .build();
}

```

```
// Replace the IAM instance profile association for the EC2 instance.
ReplaceIamInstanceProfileAssociationRequest replaceRequest =
ReplaceIamInstanceProfileAssociationRequest
    .builder()
    .iamInstanceProfile(iamInstanceProfile)
    .associationId(profileAssociationId) // Make sure
'profileAssociationId' is a valid association ID.
    .build();

try {
    getEc2Client().replaceIamInstanceProfileAssociation(replaceRequest);
    // Handle the response as needed.
} catch (Ec2Exception e) {
    // Handle exceptions, log, or report the error.
    System.err.println("Error: " + e.getMessage());
}

System.out.format("Replaced instance profile for association %s with profile
%s.", profileAssociationId,
    newInstanceProfileName);
TimeUnit.SECONDS.sleep(15);
boolean instReady = false;
int tries = 0;

// Reboot after 60 seconds
while (!instReady) {
    if (tries % 6 == 0) {
        getEc2Client().rebootInstances(RebootInstancesRequest.builder()
            .instanceIds(instanceId)
            .build());
        System.out.println("Rebooting instance " + instanceId + " and
waiting for it to be ready.");
    }
    tries++;
    try {
        TimeUnit.SECONDS.sleep(10);
    } catch (InterruptedException e) {
        e.printStackTrace();
    }

    DescribeInstanceInformationResponse informationResponse =
getSSMClient().describeInstanceInformation();
    List<InstanceInformation> instanceInformationList =
informationResponse.instanceInformationList();
    for (InstanceInformation info : instanceInformationList) {
```



```

        if (info.instanceId().equals(instanceId)) {
            instReady = true;
            break;
        }
    }
}

SendCommandRequest sendCommandRequest = SendCommandRequest.builder()
    .instanceIds(instanceId)
    .documentName("AWS-RunShellScript")
    .parameters(Collections.singletonMap("commands",
        Collections.singletonList("cd / && sudo python3 server.py
80")))
    .build();

getSSMClient().sendCommand(sendCommandRequest);
System.out.println("Restarted the Python web server on instance " +
instanceId + ".");
}

public void openInboundPort(String secGroupId, String port, String ipAddress) {
    AuthorizeSecurityGroupIngressRequest ingressRequest =
AuthorizeSecurityGroupIngressRequest.builder()
        .groupName(secGroupId)
        .cidrIp(ipAddress)
        .fromPort(Integer.parseInt(port))
        .build();

    getEc2Client().authorizeSecurityGroupIngress(ingressRequest);
    System.out.format("Authorized ingress to %s on port %s from %s.",
secGroupId, port, ipAddress);
}

/**
 * Detaches a role from an instance profile, detaches policies from the role,
 * and deletes all the resources.
 */
public void deleteInstanceProfile(String roleName, String profileName) {
    try {
        software.amazon.awssdk.services.iam.model.GetInstanceProfileRequest
getInstanceProfileRequest =
software.amazon.awssdk.services.iam.model.GetInstanceProfileRequest
        .builder()
        .instanceProfileName(profileName)

```

```
        .build();

        GetInstanceProfileResponse response =
getIAMClient().getInstanceProfile(getInstanceProfileRequest);
        String name = response.getInstanceProfile().getInstanceProfileName();
        System.out.println(name);

        RemoveRoleFromInstanceProfileRequest profileRequest =
RemoveRoleFromInstanceProfileRequest.builder()
            .instanceProfileName(profileName)
            .roleName(roleName)
            .build();

        getIAMClient().removeRoleFromInstanceProfile(profileRequest);
        DeleteInstanceProfileRequest deleteInstanceProfileRequest =
DeleteInstanceProfileRequest.builder()
            .instanceProfileName(profileName)
            .build();

        getIAMClient().deleteInstanceProfile(deleteInstanceProfileRequest);
        System.out.println("Deleted instance profile " + profileName);

        DeleteRoleRequest deleteRoleRequest = DeleteRoleRequest.builder()
            .roleName(roleName)
            .build();

        // List attached role policies.
        ListAttachedRolePoliciesResponse rolesResponse = getIAMClient()
            .listAttachedRolePolicies(role -> role.roleName(roleName));
        List<AttachedPolicy> attachedPolicies =
rolesResponse.attachedPolicies();
        for (AttachedPolicy attachedPolicy : attachedPolicies) {
            DetachRolePolicyRequest request = DetachRolePolicyRequest.builder()
                .roleName(roleName)
                .policyArn(attachedPolicy.policyArn())
                .build();

            getIAMClient().detachRolePolicy(request);
            System.out.println("Detached and deleted policy " +
attachedPolicy.policyName());
        }

        getIAMClient().deleteRole(deleteRoleRequest);
        System.out.println("Instance profile and role deleted.");
```

```
        } catch (IamException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public void deleteTemplate(String templateName) {
        getEc2Client().deleteLaunchTemplate(name ->
name.launchTemplateName(templateName));
        System.out.format(templateName + " was deleted.");
    }

    public void deleteAutoScaleGroup(String groupName) {
        DeleteAutoScalingGroupRequest deleteAutoScalingGroupRequest =
DeleteAutoScalingGroupRequest.builder()
            .autoScalingGroupName(groupName)
            .forceDelete(true)
            .build();

getAutoScalingClient().deleteAutoScalingGroup(deleteAutoScalingGroupRequest);
        System.out.println(groupName + " was deleted.");
    }

    /**
     * Verify the default security group of the specified VPC allows ingress from
     * this
     * computer. This can be done by allowing ingress from this computer's IP
     * address. In some situations, such as connecting from a corporate network, you
     * must instead specify a prefix list ID. You can also temporarily open the port
     * to
     * any IP address while running this example. If you do, be sure to remove
     * public
     * access when you're done.
     */
    public GroupInfo verifyInboundPort(String VPC, int port, String ipAddress) {
        boolean portIsOpen = false;
        GroupInfo groupInfo = new GroupInfo();
        try {
            Filter filter = Filter.builder()
                .name("group-name")
                .values("default")

```

```

        .build();

    Filter filter1 = Filter.builder()
        .name("vpc-id")
        .values(VPC)
        .build();

    DescribeSecurityGroupsRequest securityGroupsRequest =
DescribeSecurityGroupsRequest.builder()
        .filters(filter, filter1)
        .build();

    DescribeSecurityGroupsResponse securityGroupsResponse = getEc2Client()
        .describeSecurityGroups(securityGroupsRequest);
    String securityGroup =
securityGroupsResponse.securityGroups().get(0).groupName();
    groupInfo.setGroupName(securityGroup);

    for (SecurityGroup secGroup : securityGroupsResponse.securityGroups()) {
        System.out.println("Found security group: " + secGroup.groupId());

        for (IpPermission ipPermission : secGroup.ipPermissions()) {
            if (ipPermission.fromPort() == port) {
                System.out.println("Found inbound rule: " + ipPermission);
                for (IpRange ipRange : ipPermission.ipRanges()) {
                    String cidrIp = ipRange.cidrIp();
                    if (cidrIp.startsWith(ipAddress) ||
cidrIp.equals("0.0.0.0/0")) {
                        System.out.println(cidrIp + " is applicable");
                        portIsOpen = true;
                    }
                }

                if (!ipPermission.prefixListIds().isEmpty()) {
                    System.out.println("Prefix lList is applicable");
                    portIsOpen = true;
                }

                if (!portIsOpen) {
                    System.out
                        .println("The inbound rule does not appear to be
open to either this computer's IP,"
                                + " all IP addresses (0.0.0.0/0), or to
a prefix list ID.");
                }
            }
        }
    }
}

```

```
        } else {
            break;
        }
    }
}

} catch (AutoScalingException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
}

groupInfo.setPortOpen(portIsOpen);
return groupInfo;
}

/*
 * Attaches an Elastic Load Balancing (ELB) target group to this EC2 Auto
 * Scaling group.
 * The target group specifies how the load balancer forward requests to the
 * instances
 * in the group.
 */
public void attachLoadBalancerTargetGroup(String asGroupName, String
targetGroupARN) {
    try {
        AttachLoadBalancerTargetGroupsRequest targetGroupsRequest =
AttachLoadBalancerTargetGroupsRequest.builder()
            .autoScalingGroupName(asGroupName)
            .targetGroupARNs(targetGroupARN)
            .build();

getAutoScalingClient().attachLoadBalancerTargetGroups(targetGroupsRequest);
        System.out.println("Attached load balancer to " + asGroupName);

    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

// Creates an EC2 Auto Scaling group with the specified size.
public String[] createGroup(int groupSize, String templateName, String
autoScalingGroupName) {
```

```
// Get availability zones.
software.amazon.awssdk.services.ec2.model.DescribeAvailabilityZonesRequest
zonesRequest =
software.amazon.awssdk.services.ec2.model.DescribeAvailabilityZonesRequest
    .builder()
    .build();

DescribeAvailabilityZonesResponse zonesResponse =
getEc2Client().describeAvailabilityZones(zonesRequest);
List<String> availabilityZoneNames =
zonesResponse.availabilityZones().stream()

.map(software.amazon.awssdk.services.ec2.model.AvailabilityZone::zoneName)
    .collect(Collectors.toList());

String availabilityZones = String.join(",", availabilityZoneNames);
LaunchTemplateSpecification specification =
LaunchTemplateSpecification.builder()
    .launchTemplateName(templateName)
    .version("$Default")
    .build();

String[] zones = availabilityZones.split(",");
CreateAutoScalingGroupRequest groupRequest =
CreateAutoScalingGroupRequest.builder()
    .launchTemplate(specification)
    .availabilityZones(zones)
    .maxSize(groupSize)
    .minSize(groupSize)
    .autoScalingGroupName(autoScalingGroupName)
    .build();

try {
    getAutoScalingClient().createAutoScalingGroup(groupRequest);

} catch (AutoScalingException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}

System.out.println("Created an EC2 Auto Scaling group named " +
autoScalingGroupName);
return zones;
}
```

```
public String getDefaultVPC() {
    // Define the filter.
    Filter defaultFilter = Filter.builder()
        .name("is-default")
        .values("true")
        .build();

    software.amazon.awssdk.services.ec2.model.DescribeVpcsRequest request =
software.amazon.awssdk.services.ec2.model.DescribeVpcsRequest
        .builder()
        .filters(defaultFilter)
        .build();

    DescribeVpcsResponse response = getEc2Client().describeVpcs(request);
    return response.vpcs().get(0).vpcId();
}

// Gets the default subnets in a VPC for a specified list of Availability Zones.
public List<Subnet> getSubnets(String vpcId, String[] availabilityZones) {
    List<Subnet> subnets = null;
    Filter vpcFilter = Filter.builder()
        .name("vpc-id")
        .values(vpcId)
        .build();

    Filter azFilter = Filter.builder()
        .name("availability-zone")
        .values(availabilityZones)
        .build();

    Filter defaultForAZ = Filter.builder()
        .name("default-for-az")
        .values("true")
        .build();

    DescribeSubnetsRequest request = DescribeSubnetsRequest.builder()
        .filters(vpcFilter, azFilter, defaultForAZ)
        .build();

    DescribeSubnetsResponse response = getEc2Client().describeSubnets(request);
    subnets = response.subnets();
    return subnets;
}
```

```
// Gets data about the instances in the EC2 Auto Scaling group.
public String getBadInstance(String groupName) {
    DescribeAutoScalingGroupsRequest request =
DescribeAutoScalingGroupsRequest.builder()
        .autoScalingGroupNames(groupName)
        .build();

    DescribeAutoScalingGroupsResponse response =
getAutoScalingClient().describeAutoScalingGroups(request);
    AutoScalingGroup autoScalingGroup = response.autoScalingGroups().get(0);
    List<String> instanceIds = autoScalingGroup.instances().stream()
        .map(instance -> instance.instanceId())
        .collect(Collectors.toList());

    String[] instanceIdArray = instanceIds.toArray(new String[0]);
    for (String instanceId : instanceIdArray) {
        System.out.println("Instance ID: " + instanceId);
        return instanceId;
    }
    return "";
}

// Gets data about the profile associated with an instance.
public String getInstanceProfile(String instanceId) {
    Filter filter = Filter.builder()
        .name("instance-id")
        .values(instanceId)
        .build();

    DescribeIamInstanceProfileAssociationsRequest associationsRequest =
DescribeIamInstanceProfileAssociationsRequest
        .builder()
        .filters(filter)
        .build();

    DescribeIamInstanceProfileAssociationsResponse response = getEc2Client()
        .describeIamInstanceProfileAssociations(associationsRequest);
    return response.iamInstanceProfileAssociations().get(0).associationId();
}

public void deleteRolesPolicies(String policyName, String roleName, String
InstanceProfile) {
```



```

        ListPoliciesRequest listPoliciesRequest =
ListPoliciesRequest.builder().build();
        ListPoliciesResponse listPoliciesResponse =
getIAMClient().listPolicies(listPoliciesRequest);
        for (Policy policy : listPoliciesResponse.policies()) {
            if (policy.policyName().equals(policyName)) {
                // List the entities (users, groups, roles) that are attached to the
policy.

software.amazon.awssdk.services.iam.model.ListEntitiesForPolicyRequest
listEntitiesRequest =
software.amazon.awssdk.services.iam.model.ListEntitiesForPolicyRequest
                .builder()
                .policyArn(policy.arn())
                .build();
                ListEntitiesForPolicyResponse listEntitiesResponse = iamClient
                .listEntitiesForPolicy(listEntitiesRequest);
                if (!listEntitiesResponse.policyGroups().isEmpty() || !
listEntitiesResponse.policyUsers().isEmpty()
                || !listEntitiesResponse.policyRoles().isEmpty()) {
                    // Detach the policy from any entities it is attached to.
                    DetachRolePolicyRequest detachPolicyRequest =
DetachRolePolicyRequest.builder()
                        .policyArn(policy.arn())
                        .roleName(roleName) // Specify the name of the IAM role
                        .build();

                    getIAMClient().detachRolePolicy(detachPolicyRequest);
                    System.out.println("Policy detached from entities.");
                }

                // Now, you can delete the policy.
                DeletePolicyRequest deletePolicyRequest =
DeletePolicyRequest.builder()
                    .policyArn(policy.arn())
                    .build();

                getIAMClient().deletePolicy(deletePolicyRequest);
                System.out.println("Policy deleted successfully.");
                break;
            }
        }

// List the roles associated with the instance profile

```

```

        ListInstanceProfilesForRoleRequest listRolesRequest =
ListInstanceProfilesForRoleRequest.builder()
        .roleName(roleName)
        .build();

        // Detach the roles from the instance profile
        ListInstanceProfilesForRoleResponse listRolesResponse =
iamClient.listInstanceProfilesForRole(listRolesRequest);
        for (software.amazon.awssdk.services.iam.model.InstanceProfile profile :
listRolesResponse.instanceProfiles()) {
            RemoveRoleFromInstanceProfileRequest removeRoleRequest =
RemoveRoleFromInstanceProfileRequest.builder()
                .instanceProfileName(InstanceProfile)
                .roleName(roleName) // Remove the extra dot here
                .build();

            getIAMClient().removeRoleFromInstanceProfile(removeRoleRequest);
            System.out.println("Role " + roleName + " removed from instance profile
" + InstanceProfile);
        }

        // Delete the instance profile after removing all roles
        DeleteInstanceProfileRequest deleteInstanceProfileRequest =
DeleteInstanceProfileRequest.builder()
            .instanceProfileName(InstanceProfile)
            .build();

        getIAMClient().deleteInstanceProfile(r ->
r.instanceProfileName(InstanceProfile));
        System.out.println(InstanceProfile + " Deleted");
        System.out.println("All roles and policies are deleted.");
    }
}

```

Crea una classe che racchiuda le operazioni di Elastic Load Balancing.

```

public class LoadBalancer {
    public ElasticLoadBalancingV2Client elasticLoadBalancingV2Client;

    public ElasticLoadBalancingV2Client getLoadBalancerClient() {
        if (elasticLoadBalancingV2Client == null) {
            elasticLoadBalancingV2Client = ElasticLoadBalancingV2Client.builder()

```

```
        .region(Region.US_EAST_1)
        .build();
    }

    return elasticLoadBalancingV2Client;
}

// Checks the health of the instances in the target group.
public List<TargetHealthDescription> checkTargetHealth(String targetGroupName) {
    DescribeTargetGroupsRequest targetGroupsRequest =
DescribeTargetGroupsRequest.builder()
        .names(targetGroupName)
        .build();

    DescribeTargetGroupsResponse tgResponse =
getLoadBalancerClient().describeTargetGroups(targetGroupsRequest);

    DescribeTargetHealthRequest healthRequest =
DescribeTargetHealthRequest.builder()
        .targetGroupArn(tgResponse.targetGroups().get(0).targetGroupArn())
        .build();

    DescribeTargetHealthResponse healthResponse =
getLoadBalancerClient().describeTargetHealth(healthRequest);
    return healthResponse.targetHealthDescriptions();
}

// Gets the HTTP endpoint of the load balancer.
public String getEndpoint(String lbName) {
    DescribeLoadBalancersResponse res = getLoadBalancerClient()
        .describeLoadBalancers(describe -> describe.names(lbName));
    return res.loadBalancers().get(0).dnsName();
}

// Deletes a load balancer.
public void deleteLoadBalancer(String lbName) {
    try {
        // Use a waiter to delete the Load Balancer.
        DescribeLoadBalancersResponse res = getLoadBalancerClient()
            .describeLoadBalancers(describe -> describe.names(lbName));
        ElasticLoadBalancingV2Waiter loadBalancerWaiter =
getLoadBalancerClient().waiter();
        DescribeLoadBalancersRequest request =
DescribeLoadBalancersRequest.builder()
```

```

        .loadBalancerArns(res.loadBalancers().get(0).loadBalancerArn())
        .build();

        getLoadBalancerClient().deleteLoadBalancer(
            builder ->
builder.loadBalancerArn(res.loadBalancers().get(0).loadBalancerArn()));
        WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =
loadBalancerWaiter
            .waitUntilLoadBalancersDeleted(request);
        waiterResponse.matched().response().ifPresent(System.out::println);

    } catch (ElasticLoadBalancingV2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }
    System.out.println(lbName + " was deleted.");
}

// Deletes the target group.
public void deleteTargetGroup(String targetGroupName) {
    try {
        DescribeTargetGroupsResponse res = getLoadBalancerClient()
            .describeTargetGroups(describe ->
describe.names(targetGroupName));
        getLoadBalancerClient()
            .deleteTargetGroup(builder ->
builder.targetGroupArn(res.targetGroups().get(0).targetGroupArn()));
    } catch (ElasticLoadBalancingV2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }
    System.out.println(targetGroupName + " was deleted.");
}

// Verify this computer can successfully send a GET request to the load balancer
// endpoint.
public boolean verifyLoadBalancerEndpoint(String elbDnsName) throws IOException,
InterruptedException {
    boolean success = false;
    int retries = 3;
    CloseableHttpClient httpClient = HttpClients.createDefault();

    // Create an HTTP GET request to the ELB.
    HttpGet httpGet = new HttpGet("http://" + elbDnsName);
    try {
        while ((!success) && (retries > 0)) {

```

```
        // Execute the request and get the response.
        HttpResponse response = httpClient.execute(httpGet);
        int statusCode = response.getStatusLine().getStatusCode();
        System.out.println("HTTP Status Code: " + statusCode);
        if (statusCode == 200) {
            success = true;
        } else {
            retries--;
            System.out.println("Got connection error from load balancer
endpoint, retrying...");
            TimeUnit.SECONDS.sleep(15);
        }
    }

} catch (org.apache.http.conn.HttpHostConnectException e) {
    System.out.println(e.getMessage());
}

System.out.println("Status.." + success);
return success;
}

/*
 * Creates an Elastic Load Balancing target group. The target group specifies
 * how
 * the load balancer forward requests to instances in the group and how instance
 * health is checked.
 */
public String createTargetGroup(String protocol, int port, String vpcId, String
targetGroupName) {
    CreateTargetGroupRequest targetGroupRequest =
CreateTargetGroupRequest.builder()
        .healthCheckPath("/healthcheck")
        .healthCheckTimeoutSeconds(5)
        .port(port)
        .vpcId(vpcId)
        .name(targetGroupName)
        .protocol(protocol)
        .build();

    CreateTargetGroupResponse targetGroupResponse =
getLoadBalancerClient().createTargetGroup(targetGroupRequest);
    String targetGroupArn =
targetGroupResponse.targetGroups().get(0).targetGroupArn();
}
```

```
        String targetGroup =
targetGroupResponse.targetGroups().get(0).targetGroupName();
        System.out.println("The " + targetGroup + " was created with ARN" +
targetGroupArn);
        return targetGroupArn;
    }

    /**
     * Creates an Elastic Load Balancing load balancer that uses the specified
     * subnets
     * and forwards requests to the specified target group.
     */
    public String createLoadBalancer(List<Subnet> subnetIds, String targetGroupARN,
String lbName, int port,
        String protocol) {
        try {
            List<String> subnetIdStrings = subnetIds.stream()
                .map(Subnet::subnetId)
                .collect(Collectors.toList());

            CreateLoadBalancerRequest balancerRequest =
CreateLoadBalancerRequest.builder()
                .subnets(subnetIdStrings)
                .name(lbName)
                .scheme("internet-facing")
                .build();

            // Create and wait for the load balancer to become available.
            CreateLoadBalancerResponse lsResponse =
getLoadBalancerClient().createLoadBalancer(balancerRequest);
            String lbARN = lsResponse.loadBalancers().get(0).loadBalancerArn();

            ElasticLoadBalancingV2Waiter loadBalancerWaiter =
getLoadBalancerClient().waiter();
            DescribeLoadBalancersRequest request =
DescribeLoadBalancersRequest.builder()
                .loadBalancerArns(lbARN)
                .build();

            System.out.println("Waiting for Load Balancer " + lbName + " to become
available.");
            WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =
loadBalancerWaiter
                .waitUntilLoadBalancerAvailable(request);
```

```

waiterResponse.matched().response().ifPresent(System.out::println);
System.out.println("Load Balancer " + lbName + " is available.");

// Get the DNS name (endpoint) of the load balancer.
String lbDNSName = lsResponse.loadBalancers().get(0).dnsName();
System.out.println("*** Load Balancer DNS Name: " + lbDNSName);

// Create a listener for the load balance.
Action action = Action.builder()
    .targetGroupArn(targetGroupARN)
    .type("forward")
    .build();

CreateListenerRequest listenerRequest = CreateListenerRequest.builder()

.loadBalancerArn(lsResponse.loadBalancers().get(0).loadBalancerArn())
    .defaultActions(action)
    .port(port)
    .protocol(protocol)
    .defaultActions(action)
    .build();

getLoadBalancerClient().createListener(listenerRequest);
System.out.println("Created listener to forward traffic from load
balancer " + lbName + " to target group "
    + targetGroupARN);

// Return the load balancer DNS name.
return lbDNSName;

} catch (ElasticLoadBalancingV2Exception e) {
    e.printStackTrace();
}
return "";
}
}

```

Crea una classe che utilizzi DynamoDB per simulare un servizio di raccomandazione.

```

public class Database {

    private static DynamoDbClient dynamoDbClient;

```

```
public static DynamoDbClient getDynamoDbClient() {
    if (dynamoDbClient == null) {
        dynamoDbClient = DynamoDbClient.builder()
            .region(Region.US_EAST_1)
            .build();
    }
    return dynamoDbClient;
}

// Checks to see if the Amazon DynamoDB table exists.
private boolean doesTableExist(String tableName) {
    try {
        // Describe the table and catch any exceptions.
        DescribeTableRequest describeTableRequest =
DescribeTableRequest.builder()
            .tableName(tableName)
            .build();

        getDynamoDbClient().describeTable(describeTableRequest);
        System.out.println("Table '" + tableName + "' exists.");
        return true;
    } catch (ResourceNotFoundException e) {
        System.out.println("Table '" + tableName + "' does not exist.");
    } catch (DynamoDbException e) {
        System.err.println("Error checking table existence: " + e.getMessage());
    }
    return false;
}

/**
 * Creates a DynamoDB table to use a recommendation service. The table has a
 * hash key named 'MediaType' that defines the type of media recommended, such
 * as
 * Book or Movie, and a range key named 'ItemId' that, combined with the
 * MediaType,
 * forms a unique identifier for the recommended item.
 */
public void createTable(String tableName, String fileName) throws IOException {
    // First check to see if the table exists.
    boolean doesExist = doesTableExist(tableName);
    if (!doesExist) {
        DynamoDbWaiter dbWaiter = getDynamoDbClient().waiter();
    }
}
```



```

        CreateTableRequest createTableRequest = CreateTableRequest.builder()
            .tableName(tableName)
            .attributeDefinitions(
                AttributeDefinition.builder()
                    .attributeName("MediaType")
                    .attributeType(ScalarAttributeType.S)
                    .build(),
                AttributeDefinition.builder()
                    .attributeName("ItemId")
                    .attributeType(ScalarAttributeType.N)
                    .build())
            .keySchema(
                KeySchemaElement.builder()
                    .attributeName("MediaType")
                    .keyType(KeyType.HASH)
                    .build(),
                KeySchemaElement.builder()
                    .attributeName("ItemId")
                    .keyType(KeyType.RANGE)
                    .build())
            .provisionedThroughput(
                ProvisionedThroughput.builder()
                    .readCapacityUnits(5L)
                    .writeCapacityUnits(5L)
                    .build())
            .build();

        getDynamoDbClient().createTable(createTableRequest);
        System.out.println("Creating table " + tableName + "...");

        // Wait until the Amazon DynamoDB table is created.
        DescribeTableRequest tableRequest = DescribeTableRequest.builder()
            .tableName(tableName)
            .build();

        WaiterResponse<DescribeTableResponse> waiterResponse =
        dbWaiter.waitUntilTableExists(tableRequest);
        waiterResponse.matched().response().ifPresent(System.out::println);
        System.out.println("Table " + tableName + " created.");

        // Add records to the table.
        populateTable(fileName, tableName);
    }
}

```

```

public void deleteTable(String tableName) {
    getDynamoDbClient().deleteTable(table -> table.tableName(tableName));
    System.out.println("Table " + tableName + " deleted.");
}

// Populates the table with data located in a JSON file using the DynamoDB
// enhanced client.
public void populateTable(String fileName, String tableName) throws IOException
{
    DynamoDbEnhancedClient enhancedClient = DynamoDbEnhancedClient.builder()
        .dynamoDbClient(getDynamoDbClient())
        .build();
    ObjectMapper objectMapper = new ObjectMapper();
    File jsonFile = new File(fileName);
    JsonNode rootNode = objectMapper.readTree(jsonFile);

    DynamoDbTable<Recommendation> mappedTable = enhancedClient.table(tableName,
        TableSchema.fromBean(Recommendation.class));
    for (JsonNode currentNode : rootNode) {
        String mediaType = currentNode.path("MediaType").path("S").asText();
        int itemId = currentNode.path("ItemId").path("N").asInt();
        String title = currentNode.path("Title").path("S").asText();
        String creator = currentNode.path("Creator").path("S").asText();

        // Create a Recommendation object and set its properties.
        Recommendation rec = new Recommendation();
        rec.setMediaType(mediaType);
        rec.setItemId(itemId);
        rec.setTitle(title);
        rec.setCreator(creator);

        // Put the item into the DynamoDB table.
        mappedTable.putItem(rec); // Add the Recommendation to the list.
    }
    System.out.println("Added all records to the " + tableName);
}
}

```

Crea una classe che racchiuda le operazioni di Systems Manager.

```
public class ParameterHelper {
```

```
String tableName = "doc-example-resilient-architecture-table";
String dyntable = "doc-example-recommendation-service";
String failureResponse = "doc-example-resilient-architecture-failure-response";
String healthCheck = "doc-example-resilient-architecture-health-check";

public void reset() {
    put(dyntable, tableName);
    put(failureResponse, "none");
    put(healthCheck, "shallow");
}

public void put(String name, String value) {
    SsmClient ssmClient = SsmClient.builder()
        .region(Region.US_EAST_1)
        .build();

    PutParameterRequest parameterRequest = PutParameterRequest.builder()
        .name(name)
        .value(value)
        .overwrite(true)
        .type("String")
        .build();

    ssmClient.putParameter(parameterRequest);
    System.out.printf("Setting demo parameter %s to '%s'.", name, value);
}
}
```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella Documentazione di riferimento delle API AWS SDK for Java 2.x .
  - [AttachLoadBalancerTargetGroups](#)
  - [CreateAutoScalingGroup](#)
  - [CreateInstanceProfile](#)
  - [CreateLaunchTemplate](#)
  - [CreateListener](#)
  - [CreateLoadBalancer](#)
  - [CreateTargetGroup](#)
  - [DeleteAutoScalingGroup](#)


- [DeleteInstanceProfile](#)
- [DeleteLaunchTemplate](#)
- [DeleteLoadBalancer](#)
- [DeleteTargetGroup](#)
- [DescribeAutoScalingGroups](#)
- [DescribeAvailabilityZones](#)
- [DescribeIamInstanceProfileAssociations](#)
- [DescribeInstances](#)
- [DescribeLoadBalancers](#)
- [DescribeSubnets](#)
- [DescribeTargetGroups](#)
- [DescribeTargetHealth](#)
- [DescribeVpcs](#)
- [RebootInstances](#)
- [ReplaceIamInstanceProfileAssociation](#)
- [TerminateInstanceInAutoScalingGroup](#)
- [UpdateAutoScalingGroup](#)

## Nozioni di base sulle istanze

L'esempio di codice seguente mostra come:

- Creare una coppia di chiavi e un gruppo di sicurezza.
- Selezionare un'Amazon Machine Image (AMI) e un tipo di istanza compatibile e quindi creare un'istanza.
- Arrestare e riavviare l'istanza.
- Associazione di un indirizzo IP elastico all'istanza
- Connettiti alla tua istanza con SSH, quindi elimina le risorse.

## SDK per Java 2.x

 Note

C'è dell'altro GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
/**
 * Before running this Java (v2) code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * This Java example performs the following tasks:
 *
 * 1. Creates an RSA key pair and saves the private key data as a .pem file.
 * 2. Lists key pairs.
 * 3. Creates a security group for the default VPC.
 * 4. Displays security group information.
 * 5. Gets a list of Amazon Linux 2 AMIs and selects one.
 * 6. Gets more information about the image.
 * 7. Gets a list of instance types that are compatible with the selected AMI's
 * architecture.
 * 8. Creates an instance with the key pair, security group, AMI, and an
 * instance type.
 * 9. Displays information about the instance.
 * 10. Stops the instance and waits for it to stop.
 * 11. Starts the instance and waits for it to start.
 * 12. Allocates an Elastic IP address and associates it with the instance.
 * 13. Displays SSH connection info for the instance.
 * 14. Disassociates and deletes the Elastic IP address.
 * 15. Terminates the instance and waits for it to terminate.
 * 16. Deletes the security group.
 * 17. Deletes the key pair.
 */
public class EC2Scenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    public static void main(String[] args) throws InterruptedException {
```

```

final String usage = ""

    Usage:
        <keyName> <fileName> <groupName> <groupDesc> <vpcId>

    Where:
        keyName - A key pair name (for example, TestKeyPair).\s
        fileName - A file name where the key information is written to.
\s
        groupName - The name of the security group.\s
        groupDesc - The description of the security group.\s
        vpcId - A VPC Id value. You can get this value from the AWS
Management Console.\s
        myIpAddress - The IP address of your development machine.\s

    """;

if (args.length != 6) {
    System.out.println(usage);
    System.exit(1);
}

String keyName = args[0];
String fileName = args[1];
String groupName = args[2];
String groupDesc = args[3];
String vpcId = args[4];
String myIpAddress = args[5];

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

SsmClient ssmClient = SsmClient.builder()
    .region(region)
    .build();

System.out.println(DASHES);
System.out.println("Welcome to the Amazon EC2 example scenario.");
System.out.println(DASHES);

System.out.println(DASHES);

```

```
        System.out.println("1. Create an RSA key pair and save the private key
material as a .pem file.");
        createKeyPair(ec2, keyName, fileName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("2. List key pairs.");
        describeKeys(ec2);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. Create a security group.");
        String groupId = createSecurityGroup(ec2, groupName, groupDesc, vpcId,
myIpAddress);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. Display security group info for the newly created
security group.");
        describeSecurityGroups(ec2, groupId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. Get a list of Amazon Linux 2 AMIs and selects one
with amzn2 in the name.");
        String instanceId = getParaValues(ssmClient);
        System.out.println("The instance Id is " + instanceId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6. Get more information about an amzn2 image.");
        String amiValue = describeImage(ec2, instanceId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Get a list of instance types.");
        String instanceType = getInstanceTypes(ec2);
        System.out.println("The instance type is " + instanceType);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("8. Create an instance.");
        String newInstanceId = runInstance(ec2, instanceType, keyName, groupName,
amiValue);
```

```
System.out.println("The instance Id is " + newInstanceId);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("9. Display information about the running instance. ");
String ipAddress = describeEC2Instances(ec2, newInstanceId);
System.out.println("You can SSH to the instance using this command:");
System.out.println("ssh -i " + fileName + "ec2-user@" + ipAddress);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("10. Stop the instance and use a waiter.");
stopInstance(ec2, newInstanceId);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("11. Start the instance and use a waiter.");
startInstance(ec2, newInstanceId);
ipAddress = describeEC2Instances(ec2, newInstanceId);
System.out.println("You can SSH to the instance using this command:");
System.out.println("ssh -i " + fileName + "ec2-user@" + ipAddress);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("12. Allocate an Elastic IP address and associate it with
the instance.");
String allocationId = allocateAddress(ec2);
System.out.println("The allocation Id value is " + allocationId);
String associationId = associateAddress(ec2, newInstanceId, allocationId);
System.out.println("The associate Id value is " + associationId);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("13. Describe the instance again.");
ipAddress = describeEC2Instances(ec2, newInstanceId);
System.out.println("You can SSH to the instance using this command:");
System.out.println("ssh -i " + fileName + "ec2-user@" + ipAddress);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("14. Disassociate and release the Elastic IP address.");
disassociateAddress(ec2, associationId);
releaseEC2Address(ec2, allocationId);
System.out.println(DASHES);
```



```
System.out.println(DASHES);
System.out.println("15. Terminate the instance and use a waiter.");
terminateEC2(ec2, newInstanceId);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("16. Delete the security group.");
deleteEC2SecGroup(ec2, groupId);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("17. Delete the key.");
deleteKeys(ec2, keyName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("You successfully completed the Amazon EC2 scenario.");
System.out.println(DASHES);
ec2.close();
}

public static void deleteEC2SecGroup(Ec2Client ec2, String groupId) {
    try {
        DeleteSecurityGroupRequest request =
DeleteSecurityGroupRequest.builder()
        .groupId(groupId)
        .build();

        ec2.deleteSecurityGroup(request);
        System.out.println("Successfully deleted security group with Id " +
groupId);

    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void terminateEC2(Ec2Client ec2, String instanceId) {
    try {
        Ec2Waiter ec2Waiter = Ec2Waiter.builder()
        .overrideConfiguration(b -> b.maxAttempts(100))
        .client(ec2)
```

```
        .build());

    TerminateInstancesRequest ti = TerminateInstancesRequest.builder()
        .instanceIds(instanceId)
        .build();

    System.out.println("Use an Ec2Waiter to wait for the instance to
terminate. This will take a few minutes.");
    ec2.terminateInstances(ti);
    DescribeInstancesRequest instanceRequest =
DescribeInstancesRequest.builder()
        .instanceIds(instanceId)
        .build();

    WaiterResponse<DescribeInstancesResponse> waiterResponse = ec2Waiter
        .waitUntilInstanceTerminated(instanceRequest);
    waiterResponse.matched().response().ifPresent(System.out::println);
    System.out.println("Successfully started instance " + instanceId);
    System.out.println(instanceId + " is terminated!");

    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void deleteKeys(Ec2Client ec2, String keyPair) {
    try {
        DeleteKeyPairRequest request = DeleteKeyPairRequest.builder()
            .keyName(keyPair)
            .build();

        ec2.deleteKeyPair(request);
        System.out.println("Successfully deleted key pair named " + keyPair);

    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void releaseEC2Address(Ec2Client ec2, String allocId) {
    try {
        ReleaseAddressRequest request = ReleaseAddressRequest.builder()
```

```
        .allocationId(allocId)
        .build();

        ec2.releaseAddress(request);
        System.out.println("Successfully released Elastic IP address " +
allocId);
    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void disassociateAddress(Ec2Client ec2, String associationId) {
    try {
        DisassociateAddressRequest addressRequest =
DisassociateAddressRequest.builder()
            .associationId(associationId)
            .build();

        ec2.disassociateAddress(addressRequest);
        System.out.println("You successfully disassociated the address!");

    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static String associateAddress(Ec2Client ec2, String instanceId, String
allocationId) {
    try {
        AssociateAddressRequest associateRequest =
AssociateAddressRequest.builder()
            .instanceId(instanceId)
            .allocationId(allocationId)
            .build();

        AssociateAddressResponse associateResponse =
ec2.associateAddress(associateRequest);
        return associateResponse.associationId();

    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

```
    }
    return "";
}

public static String allocateAddress(Ec2Client ec2) {
    try {
        AllocateAddressRequest allocateRequest =
AllocateAddressRequest.builder()
            .domain(DomainType.VPC)
            .build();

        AllocateAddressResponse allocateResponse =
ec2.allocateAddress(allocateRequest);
        return allocateResponse.allocationId();

    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}

public static void startInstance(Ec2Client ec2, String instanceId) {
    Ec2Waiter ec2Waiter = Ec2Waiter.builder()
        .overrideConfiguration(b -> b.maxAttempts(100))
        .client(ec2)
        .build();

    StartInstancesRequest request = StartInstancesRequest.builder()
        .instanceIds(instanceId)
        .build();

    System.out.println("Use an Ec2Waiter to wait for the instance to run. This
will take a few minutes.");
    ec2.startInstances(request);
    DescribeInstancesRequest instanceRequest =
DescribeInstancesRequest.builder()
        .instanceIds(instanceId)
        .build();

    WaiterResponse<DescribeInstancesResponse> waiterResponse =
ec2Waiter.waitUntilInstanceRunning(instanceRequest);
    waiterResponse.matched().response().ifPresent(System.out::println);
    System.out.println("Successfully started instance " + instanceId);
}
```

```

}

public static void stopInstance(Ec2Client ec2, String instanceId) {
    Ec2Waiter ec2Waiter = Ec2Waiter.builder()
        .overrideConfiguration(b -> b.maxAttempts(100))
        .client(ec2)
        .build();
    StopInstancesRequest request = StopInstancesRequest.builder()
        .instanceIds(instanceId)
        .build();

    System.out.println("Use an Ec2Waiter to wait for the instance to stop. This
will take a few minutes.");
    ec2.stopInstances(request);
    DescribeInstancesRequest instanceRequest =
DescribeInstancesRequest.builder()
        .instanceIds(instanceId)
        .build();

    WaiterResponse<DescribeInstancesResponse> waiterResponse =
ec2Waiter.waitUntilInstanceStopped(instanceRequest);
    waiterResponse.matched().response().ifPresent(System.out::println);
    System.out.println("Successfully stopped instance " + instanceId);
}

public static String describeEC2Instances(Ec2Client ec2, String newInstanceId) {
    try {
        String pubAddress = "";
        boolean isRunning = false;
        DescribeInstancesRequest request = DescribeInstancesRequest.builder()
            .instanceIds(newInstanceId)
            .build();

        while (!isRunning) {
            DescribeInstancesResponse response = ec2.describeInstances(request);
            String state =
response.reservations().get(0).instances().get(0).state().name().name();
            if (state.compareTo("RUNNING") == 0) {
                System.out.println("Image id is " +
response.reservations().get(0).instances().get(0).imageId());
                System.out.println(
                    "Instance type is " +
response.reservations().get(0).instances().get(0).instanceType());
                System.out.println(

```

```

        "Instance state is " +
response.reservations().get(0).instances().get(0).state().name());
        pubAddress =
response.reservations().get(0).instances().get(0).publicIpAddress();
        System.out.println("Instance address is " + pubAddress);
        isRunning = true;
    }
}
return pubAddress;
} catch (SsmException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
return "";
}

public static String runInstance(Ec2Client ec2, String instanceType, String
keyName, String groupName,
    String amiId) {
    try {
        RunInstancesRequest runRequest = RunInstancesRequest.builder()
            .instanceType(instanceType)
            .keyName(keyName)
            .securityGroups(groupName)
            .maxCount(1)
            .minCount(1)
            .imageId(amiId)
            .build();

        System.out.println("Going to start an EC2 instance using a waiter");
        RunInstancesResponse response = ec2.runInstances(runRequest);
        String instanceIdVal = response.instances().get(0).instanceId();
        ec2.waiter().waitUntilInstanceRunning(r ->
r.instanceIds(instanceIdVal));
        System.out.println("Successfully started EC2 instance " + instanceIdVal
+ " based on AMI " + amiId);
        return instanceIdVal;

    } catch (SsmException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return "";
}
}

```

```
// Get a list of instance types.
public static String getInstanceTypes(Ec2Client ec2) {
    String instanceType;
    try {
        DescribeInstanceTypesRequest typesRequest =
DescribeInstanceTypesRequest.builder()
        .maxResults(10)
        .build();

        DescribeInstanceTypesResponse response =
ec2.describeInstanceTypes(typesRequest);
        List<InstanceTypeInfo> instanceTypes = response.instanceTypes();
        for (InstanceTypeInfo type : instanceTypes) {
            System.out.println("The memory information of this type is " +
type.memoryInfo().sizeInMiB());
            System.out.println("Network information is " +
type.networkInfo().toString());
            System.out.println("Instance type is " +
type.instanceType().toString());
            instanceType = type.instanceType().toString();
            if (instanceType.compareTo("t2.2xlarge") == 0){
                return instanceType;
            }
        }

    } catch (SsmException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return "";
}

// Display the Description field that corresponds to the instance Id value.
public static String describeImage(Ec2Client ec2, String instanceId) {
    try {
        DescribeImagesRequest imagesRequest = DescribeImagesRequest.builder()
        .imageIds(instanceId)
        .build();

        DescribeImagesResponse response = ec2.describeImages(imagesRequest);
        System.out.println("The description of the first image is " +
response.images().get(0).description());
    }
}
```

```
        System.out.println("The name of the first image is " +
response.images().get(0).name());

        // Return the image Id value.
        return response.images().get(0).imageId();

    } catch (SsmException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return "";
}

// Get the Id value of an instance with amzn2 in the name.
public static String getParaValues(SsmClient ssmClient) {
    try {
        GetParametersByPathRequest parameterRequest =
GetParametersByPathRequest.builder()
            .path("/aws/service/ami-amazon-linux-latest")
            .build();

        GetParametersByPathIterable responses =
ssmClient.getParametersByPathPaginator(parameterRequest);
        for
(ssoftware.amazon.awssdk.services.ssm.model.GetParametersByPathResponse response :
responses) {
            System.out.println("Test " + response.nextToken());
            List<Parameter> parameterList = response.parameters();
            for (Parameter para : parameterList) {
                System.out.println("The name of the para is: " + para.name());
                System.out.println("The type of the para is: " + para.type());
                if (filterName(para.name())) {
                    return para.value();
                }
            }
        }

    } catch (SsmException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return "";
}
```



```
// Return true if the name has amzn2 in it. For example:
// /aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-arm64-gp2
private static boolean filterName(String name) {
    String[] parts = name.split("/");
    String myValue = parts[4];
    return myValue.contains("amzn2");
}

public static void describeSecurityGroups(Ec2Client ec2, String groupId) {
    try {
        DescribeSecurityGroupsRequest request =
DescribeSecurityGroupsRequest.builder()
            .groupIds(groupId)
            .build();

        // Use a paginator.
        DescribeSecurityGroupsIterable listGroups =
ec2.describeSecurityGroupsPaginator(request);
        listGroups.stream()
            .flatMap(r -> r.securityGroups().stream())
            .forEach(group -> System.out
                .println(" Group id: " +group.groupId() + " group name = " +
group.groupName()));

    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static String createSecurityGroup(Ec2Client ec2, String groupName, String
groupDesc, String vpcId,
    String myIpAddress) {
    try {
        CreateSecurityGroupRequest createRequest =
CreateSecurityGroupRequest.builder()
            .groupName(groupName)
            .description(groupDesc)
            .vpcId(vpcId)
            .build();

        CreateSecurityGroupResponse resp =
ec2.createSecurityGroup(createRequest);
        IpRange ipRange = IpRange.builder()
```

```
        .cidrIp(myIpAddress + "/0")
        .build();

    IpPermission ipPerm = IpPermission.builder()
        .ipProtocol("tcp")
        .toPort(80)
        .fromPort(80)
        .ipRanges(ipRange)
        .build();

    IpPermission ipPerm2 = IpPermission.builder()
        .ipProtocol("tcp")
        .toPort(22)
        .fromPort(22)
        .ipRanges(ipRange)
        .build();

    AuthorizeSecurityGroupIngressRequest authRequest =
AuthorizeSecurityGroupIngressRequest.builder()
        .groupName(groupName)
        .ipPermissions(ipPerm, ipPerm2)
        .build();

    ec2.authorizeSecurityGroupIngress(authRequest);
    System.out.println("Successfully added ingress policy to security group
" + groupName);
    return resp.groupId();

    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}

public static void describeKeys(Ec2Client ec2) {
    try {
        DescribeKeyPairsResponse response = ec2.describeKeyPairs();
        response.keyPairs().forEach(keyPair -> System.out.printf(
            "Found key pair with name %s " +
                "and fingerprint %s",
            keyPair.keyName(),
            keyPair.keyFingerprint()));
    }
```

```
        } catch (Ec2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void createKeyPair(Ec2Client ec2, String keyName, String fileName)
    {
        try {
            CreateKeyPairRequest request = CreateKeyPairRequest.builder()
                .keyName(keyName)
                .build();

            CreateKeyPairResponse response = ec2.createKeyPair(request);
            String content = response.keyMaterial();
            BufferedWriter writer = new BufferedWriter(new FileWriter(fileName));
            writer.write(content);
            writer.close();
            System.out.println("Successfully created key pair named " + keyName);

        } catch (Ec2Exception | IOException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella Documentazione di riferimento delle API AWS SDK for Java 2.x .
  - [AllocateAddress](#)
  - [AssociateAddress](#)
  - [AuthorizeSecurityGroupIngress](#)
  - [CreateKeyPair](#)
  - [CreateSecurityGroup](#)
  - [DeleteKeyPair](#)
  - [DeleteSecurityGroup](#)
  - [DescribeImages](#)
  - [DescribeInstanceTypes](#)

- [DescribeInstances](#)
- [DescribeKeyPairs](#)
- [DescribeSecurityGroups](#)
- [DisassociateAddress](#)
- [ReleaseAddress](#)
- [RunInstances](#)
- [StartInstances](#)
- [StopInstances](#)
- [TerminateInstances](#)
- [UnmonitorInstances](#)

## Esempi di Amazon ECS con SDK for Java 2.x

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando Amazon ECS. AWS SDK for Java 2.x

Le operazioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le operazioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati e negli esempi tra servizi.

Scenari: esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio.

Ogni esempio include un collegamento a GitHub, dove puoi trovare istruzioni su come configurare ed eseguire il codice nel contesto.

Argomenti

- [Azioni](#)

Azioni

### **CreateCluster**

Il seguente esempio di codice mostra come utilizzare `CreateCluster`.

## SDK per Java 2.x

 Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ecs.EcsClient;
import software.amazon.awssdk.services.ecs.model.ExecuteCommandConfiguration;
import software.amazon.awssdk.services.ecs.model.ExecuteCommandLogging;
import software.amazon.awssdk.services.ecs.model.ClusterConfiguration;
import software.amazon.awssdk.services.ecs.model.CreateClusterResponse;
import software.amazon.awssdk.services.ecs.model.EcsException;
import software.amazon.awssdk.services.ecs.model.CreateClusterRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateCluster {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <clusterName>\s

                Where:
                clusterName - The name of the ECS cluster to create.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String clusterName = args[0];
```

```
Region region = Region.US_EAST_1;
EcsClient ecsClient = EcsClient.builder()
    .region(region)
    .build();

String clusterArn = createGivenCluster(ecsClient, clusterName);
System.out.println("The cluster ARN is " + clusterArn);
ecsClient.close();
}

public static String createGivenCluster(EcsClient ecsClient, String clusterName)
{
    try {
        ExecuteCommandConfiguration commandConfiguration =
ExecuteCommandConfiguration.builder()
            .logging(ExecuteCommandLogging.DEFAULT)
            .build();

        ClusterConfiguration clusterConfiguration =
ClusterConfiguration.builder()
            .executeCommandConfiguration(commandConfiguration)
            .build();

        CreateClusterRequest clusterRequest = CreateClusterRequest.builder()
            .clusterName(clusterName)
            .configuration(clusterConfiguration)
            .build();

        CreateClusterResponse response =
ecsClient.createCluster(clusterRequest);
        return response.cluster().clusterArn();

    } catch (EcsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
}
```

- Per i dettagli sull'API, [CreateCluster](#) consulta AWS SDK for Java 2.x API Reference.

## CreateService

Il seguente esempio di codice mostra come utilizzare `CreateService`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ecs.EcsClient;
import software.amazon.awssdk.services.ecs.model.AwsVpcConfiguration;
import software.amazon.awssdk.services.ecs.model.NetworkConfiguration;
import software.amazon.awssdk.services.ecs.model.CreateServiceRequest;
import software.amazon.awssdk.services.ecs.model.LaunchType;
import software.amazon.awssdk.services.ecs.model.CreateServiceResponse;
import software.amazon.awssdk.services.ecs.model.EcsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateService {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <clusterName> <serviceName> <securityGroups>
                <subnets> <taskDefinition>

                Where:
                clusterName - The name of the ECS cluster.
                serviceName - The name of the ECS service to
                create.

                securityGroups - The name of the security group.
                subnets - The name of the subnet.
```

```
        taskDefinition - The name of the task definition.
        """);

    if (args.length != 5) {
        System.out.println(usage);
        System.exit(1);
    }

    String clusterName = args[0];
    String serviceName = args[1];
    String securityGroups = args[2];
    String subnets = args[3];
    String taskDefinition = args[4];
    Region region = Region.US_EAST_1;
    EcsClient ecsClient = EcsClient.builder()
        .region(region)
        .build();

    String serviceArn = createNewService(ecsClient, clusterName,
serviceName, securityGroups, subnets,
        taskDefinition);
    System.out.println("The ARN of the service is " + serviceArn);
    ecsClient.close();
}

public static String createNewService(EcsClient ecsClient,
    String clusterName,
    String serviceName,
    String securityGroups,
    String subnets,
    String taskDefinition) {

    try {
        AwsVpcConfiguration vpcConfiguration =
AwsVpcConfiguration.builder()
            .securityGroups(securityGroups)
            .subnets(subnets)
            .build();

        NetworkConfiguration configuration =
NetworkConfiguration.builder()
            .awsvpcConfiguration(vpcConfiguration)
            .build();
```



```

        CreateServiceRequest serviceRequest =
CreateServiceRequest.builder()
                        .cluster(clusterName)
                        .networkConfiguration(configuration)
                        .desiredCount(1)
                        .launchType(LaunchType.FARGATE)
                        .serviceName(serviceName)
                        .taskDefinition(taskDefinition)
                        .build();

        CreateServiceResponse response =
ecsClient.createService(serviceRequest);
        return response.service().serviceArn();

    } catch (EcsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
}

```

- Per i dettagli sull'API, [CreateService](#) consulta AWS SDK for Java 2.x API Reference.

## DeleteService

Il seguente esempio di codice mostra come utilizzare `DeleteService`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ecs.EcsClient;
import software.amazon.awssdk.services.ecs.model.DeleteServiceRequest;
import software.amazon.awssdk.services.ecs.model.EcsException;

```

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class DeleteService {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <clusterName> <serviceArn>\s

            Where:
                clusterName - The name of the ECS cluster.
                serviceArn - The ARN of the ECS service.
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String clusterName = args[0];
        String serviceArn = args[1];
        Region region = Region.US_EAST_1;
        EcsClient ecsClient = EcsClient.builder()
            .region(region)
            .build();

        deleteSpecificService(ecsClient, clusterName, serviceArn);
        ecsClient.close();
    }

    public static void deleteSpecificService(EcsClient ecsClient, String
clusterName, String serviceArn) {
        try {
            DeleteServiceRequest serviceRequest = DeleteServiceRequest.builder()
                .cluster(clusterName)
                .service(serviceArn)
                .build();
        }
    }
}
```

```
        ecsClient.deleteService(serviceRequest);
        System.out.println("The Service was successfully deleted");

    } catch (EcsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Per i dettagli sull'API, [DeleteService](#) consulta AWS SDK for Java 2.x API Reference.

## DescribeClusters

Il seguente esempio di codice mostra come utilizzare `DescribeClusters`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ecs.EcsClient;
import software.amazon.awssdk.services.ecs.model.DescribeClustersRequest;
import software.amazon.awssdk.services.ecs.model.DescribeClustersResponse;
import software.amazon.awssdk.services.ecs.model.Cluster;
import software.amazon.awssdk.services.ecs.model.EcsException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
```

```
public class DescribeClusters {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
            <clusterArn> \s

            Where:
            clusterArn - The ARN of the ECS cluster to describe.
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String clusterArn = args[0];
        Region region = Region.US_EAST_1;
        EcsClient ecsClient = EcsClient.builder()
            .region(region)
            .build();

        descCluster(ecsClient, clusterArn);
    }

    public static void descCluster(EcsClient ecsClient, String clusterArn) {
        try {
            DescribeClustersRequest clustersRequest =
DescribeClustersRequest.builder()
                .clusters(clusterArn)
                .build();

            DescribeClustersResponse response =
ecsClient.describeClusters(clustersRequest);
            List<Cluster> clusters = response.clusters();
            for (Cluster cluster : clusters) {
                System.out.println("The cluster name is " + cluster.clusterName());
            }
        } catch (EcsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

```
}

```

- Per i dettagli sull'API, [DescribeClusters](#) consulta AWS SDK for Java 2.x API Reference.

## DescribeTasks

Il seguente esempio di codice mostra come utilizzare `DescribeTasks`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ecs.EcsClient;
import software.amazon.awssdk.services.ecs.model.DescribeTasksRequest;
import software.amazon.awssdk.services.ecs.model.DescribeTasksResponse;
import software.amazon.awssdk.services.ecs.model.EcsException;
import software.amazon.awssdk.services.ecs.model.Task;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListTaskDefinitions {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <clusterArn> <taskId>\s

                Where:

```

```
        clusterArn - The ARN of an ECS cluster.
        taskId - The task Id value.
        """;

    if (args.length != 2) {
        System.out.println(usage);
        System.exit(1);
    }

    String clusterArn = args[0];
    String taskId = args[1];
    Region region = Region.US_EAST_1;
    EcsClient ecsClient = EcsClient.builder()
        .region(region)
        .build();

    getAllTasks(ecsClient, clusterArn, taskId);
    ecsClient.close();
}

public static void getAllTasks(EcsClient ecsClient, String clusterArn, String
taskId) {
    try {
        DescribeTasksRequest tasksRequest = DescribeTasksRequest.builder()
            .cluster(clusterArn)
            .tasks(taskId)
            .build();

        DescribeTasksResponse response = ecsClient.describeTasks(tasksRequest);
        List<Task> tasks = response.tasks();
        for (Task task : tasks) {
            System.out.println("The task ARN is " + task.taskDefinitionArn());
        }

    } catch (EcsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Per i dettagli sull'API, [DescribeTasks](#) consulta AWS SDK for Java 2.x API Reference.

## ListClusters

Il seguente esempio di codice mostra come utilizzare `ListClusters`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ecs.EcsClient;
import software.amazon.awssdk.services.ecs.model.ListClustersResponse;
import software.amazon.awssdk.services.ecs.model.EcsException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class ListClusters {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        EcsClient ecsClient = EcsClient.builder()
            .region(region)
            .build();

        listAllClusters(ecsClient);
        ecsClient.close();
    }

    public static void listAllClusters(EcsClient ecsClient) {
        try {
            ListClustersResponse response = ecsClient.listClusters();
            List<String> clusters = response.clusterArns();
            for (String cluster : clusters) {
```

```
        System.out.println("The cluster arn is " + cluster);
    }

    } catch (EcsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Per i dettagli sull'API, [ListClusters](#) consulta AWS SDK for Java 2.x API Reference.

## UpdateService

Il seguente esempio di codice mostra come utilizzare `UpdateService`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ecs.EcsClient;
import software.amazon.awssdk.services.ecs.model.EcsException;
import software.amazon.awssdk.services.ecs.model.UpdateServiceRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class UpdateService {

    public static void main(String[] args) {
```



```
final String usage = ""

    Usage:
        <clusterName> <serviceArn>\s

    Where:
        clusterName - The cluster name.
        serviceArn - The service ARN value.
    """;

if (args.length != 2) {
    System.out.println(usage);
    System.exit(1);
}

String clusterName = args[0];
String serviceArn = args[1];
Region region = Region.US_EAST_1;
EcsClient ecsClient = EcsClient.builder()
    .region(region)
    .build();

updateSpecificService(ecsClient, clusterName, serviceArn);
ecsClient.close();
}

public static void updateSpecificService(EcsClient ecsClient, String
clusterName, String serviceArn) {
    try {
        UpdateServiceRequest serviceRequest = UpdateServiceRequest.builder()
            .cluster(clusterName)
            .service(serviceArn)
            .desiredCount(0)
            .build();

        ecsClient.updateService(serviceRequest);
        System.out.println("The service was modified");

    } catch (EcsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

```
}
```

- Per i dettagli sull'API, [UpdateService](#) consulta AWS SDK for Java 2.x API Reference.

## Elastic Load Balancing - Esempi della versione 2 che utilizzano SDK for Java 2.x

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando AWS SDK for Java 2.x with Elastic Load Balancing - Versione 2.

Le operazioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le operazioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati e negli esempi tra servizi.

Scenari: esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio.

Ogni esempio include un collegamento a GitHub, dove è possibile trovare istruzioni su come configurare ed eseguire il codice nel contesto.

Nozioni di base

Salve Elastic Load Balancing

I seguenti esempi di codice mostrano come iniziare a utilizzare Elastic Load Balancing.

SDK per Java 2.x

### Note

C'è altro su [GitHub](#) Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public class HelloLoadBalancer {  
  
    public static void main(String[] args) {  
        ElasticLoadBalancingV2Client loadBalancingV2Client =  
        ElasticLoadBalancingV2Client.builder()  
    }  
}
```

```
        .region(Region.US_EAST_1)
        .build();

        DescribeLoadBalancersResponse loadBalancersResponse =
loadBalancingV2Client
        .describeLoadBalancers(r -> r.pageSize(10));
        List<LoadBalancer> loadBalancerList =
loadBalancersResponse.loadBalancers();
        for (LoadBalancer lb : loadBalancerList)
            System.out.println("Load Balancer DNS name = " +
lb.dnsName());
    }
}
```

- Per i dettagli sull'API, [DescribeLoadBalancers](#) consulta AWS SDK for Java 2.x API Reference.

## Argomenti

- [Azioni](#)
- [Scenari](#)

## Azioni

### CreateListener

Il seguente esempio di codice mostra come utilizzare `CreateListener`.

SDK per Java 2.x

#### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
/*
 * Creates an Elastic Load Balancing load balancer that uses the specified
 * subnets
 * and forwards requests to the specified target group.
 */
```

```
public String createLoadBalancer(List<Subnet> subnetIds, String targetGroupARN,
String lbName, int port,
    String protocol) {
    try {
        List<String> subnetIdStrings = subnetIds.stream()
            .map(Subnet::subnetId)
            .collect(Collectors.toList());

        CreateLoadBalancerRequest balancerRequest =
CreateLoadBalancerRequest.builder()
            .subnets(subnetIdStrings)
            .name(lbName)
            .scheme("internet-facing")
            .build();

        // Create and wait for the load balancer to become available.
        CreateLoadBalancerResponse lsResponse =
getLoadBalancerClient().createLoadBalancer(balancerRequest);
        String lbARN = lsResponse.loadBalancers().get(0).loadBalancerArn();

        ElasticLoadBalancingV2Waiter loadBalancerWaiter =
getLoadBalancerClient().waiter();
        DescribeLoadBalancersRequest request =
DescribeLoadBalancersRequest.builder()
            .loadBalancerArns(lbARN)
            .build();

        System.out.println("Waiting for Load Balancer " + lbName + " to become
available.");
        WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =
loadBalancerWaiter
            .waitUntilLoadBalancerAvailable(request);
        waiterResponse.matched().response().ifPresent(System.out::println);
        System.out.println("Load Balancer " + lbName + " is available.");

        // Get the DNS name (endpoint) of the load balancer.
        String lbDNSName = lsResponse.loadBalancers().get(0).dnsName();
        System.out.println("*** Load Balancer DNS Name: " + lbDNSName);

        // Create a listener for the load balance.
        Action action = Action.builder()
            .targetGroupArn(targetGroupARN)
            .type("forward")
            .build();
```

```
        CreateListenerRequest listenerRequest = CreateListenerRequest.builder()

        .loadBalancerArn(lsResponse.loadBalancers().get(0).loadBalancerArn())
            .defaultActions(action)
            .port(port)
            .protocol(protocol)
            .defaultActions(action)
            .build();

        getLoadBalancerClient().createListener(listenerRequest);
        System.out.println("Created listener to forward traffic from load
balancer " + lbName + " to target group "
            + targetGroupARN);

        // Return the load balancer DNS name.
        return lbDNSName;

    } catch (ElasticLoadBalancingV2Exception e) {
        e.printStackTrace();
    }
    return "";
}
```

- Per i dettagli sull'API, [CreateListener](#) consulta AWS SDK for Java 2.x API Reference.

## CreateLoadBalancer

Il seguente esempio di codice mostra come utilizzare `CreateLoadBalancer`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
/*
 * Creates an Elastic Load Balancing load balancer that uses the specified
 * subnets
```

```
    * and forwards requests to the specified target group.
    */
    public String createLoadBalancer(List<Subnet> subnetIds, String targetGroupARN,
String lbName, int port,
    String protocol) {
        try {
            List<String> subnetIdStrings = subnetIds.stream()
                .map(Subnet::subnetId)
                .collect(Collectors.toList());

            CreateLoadBalancerRequest balancerRequest =
CreateLoadBalancerRequest.builder()
                .subnets(subnetIdStrings)
                .name(lbName)
                .scheme("internet-facing")
                .build();

            // Create and wait for the load balancer to become available.
            CreateLoadBalancerResponse lsResponse =
getLoadBalancerClient().createLoadBalancer(balancerRequest);
            String lbARN = lsResponse.loadBalancers().get(0).loadBalancerArn();

            ElasticLoadBalancingV2Waiter loadBalancerWaiter =
getLoadBalancerClient().waiter();
            DescribeLoadBalancersRequest request =
DescribeLoadBalancersRequest.builder()
                .loadBalancerArns(lbARN)
                .build();

            System.out.println("Waiting for Load Balancer " + lbName + " to become
available.");
            WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =
loadBalancerWaiter
                .waitUntilLoadBalancerAvailable(request);
            waiterResponse.matched().response().ifPresent(System.out::println);
            System.out.println("Load Balancer " + lbName + " is available.");

            // Get the DNS name (endpoint) of the load balancer.
            String lbDNSName = lsResponse.loadBalancers().get(0).dnsName();
            System.out.println("*** Load Balancer DNS Name: " + lbDNSName);

            // Create a listener for the load balance.
            Action action = Action.builder()
                .targetGroupArn(targetGroupARN)
```

```

        .type("forward")
        .build();

        CreateListenerRequest listenerRequest = CreateListenerRequest.builder()

.loadBalancerArn(lsResponse.loadBalancers().get(0).loadBalancerArn())
        .defaultActions(action)
        .port(port)
        .protocol(protocol)
        .defaultActions(action)
        .build();

        getLoadBalancerClient().createListener(listenerRequest);
        System.out.println("Created listener to forward traffic from load
balancer " + lbName + " to target group "
        + targetGroupARN);

        // Return the load balancer DNS name.
        return lbDNSName;

    } catch (ElasticLoadBalancingV2Exception e) {
        e.printStackTrace();
    }
    return "";
}

```

- Per i dettagli sull'API, [CreateLoadBalancer](#) consulta AWS SDK for Java 2.x API Reference.

## CreateTargetGroup

Il seguente esempio di codice mostra come utilizzare `CreateTargetGroup`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
/*
```

```
* Creates an Elastic Load Balancing target group. The target group specifies
* how
* the load balancer forward requests to instances in the group and how instance
* health is checked.
*/
public String createTargetGroup(String protocol, int port, String vpcId, String
targetGroupName) {
    CreateTargetGroupRequest targetGroupRequest =
CreateTargetGroupRequest.builder()
        .healthCheckPath("/healthcheck")
        .healthCheckTimeoutSeconds(5)
        .port(port)
        .vpcId(vpcId)
        .name(targetGroupName)
        .protocol(protocol)
        .build();

    CreateTargetGroupResponse targetGroupResponse =
getLoadBalancerClient().createTargetGroup(targetGroupRequest);
    String targetGroupArn =
targetGroupResponse.targetGroups().get(0).targetGroupArn();
    String targetGroup =
targetGroupResponse.targetGroups().get(0).targetGroupName();
    System.out.println("The " + targetGroup + " was created with ARN" +
targetGroupArn);
    return targetGroupArn;
}
```

- Per i dettagli sull'API, [CreateTargetGroup](#) consulta AWS SDK for Java 2.x API Reference.

## DeleteLoadBalancer

Il seguente esempio di codice mostra come utilizzare `DeleteLoadBalancer`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).



```
// Deletes a load balancer.
public void deleteLoadBalancer(String lbName) {
    try {
        // Use a waiter to delete the Load Balancer.
        DescribeLoadBalancersResponse res = getLoadBalancerClient()
            .describeLoadBalancers(describe -> describe.names(lbName));
        ElasticLoadBalancingV2Waiter loadBalancerWaiter =
getLoadBalancerClient().waiter();
        DescribeLoadBalancersRequest request =
DescribeLoadBalancersRequest.builder()
            .loadBalancerArns(res.loadBalancers().get(0).loadBalancerArn())
            .build();

        getLoadBalancerClient().deleteLoadBalancer(
            builder ->
builder.loadBalancerArn(res.loadBalancers().get(0).loadBalancerArn()));
        WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =
loadBalancerWaiter
            .waitUntilLoadBalancersDeleted(request);
        waiterResponse.matched().response().ifPresent(System.out::println);

    } catch (ElasticLoadBalancingV2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }
    System.out.println(lbName + " was deleted.");
}
```

- Per i dettagli sull'API, [DeleteLoadBalancer](#) consulta AWS SDK for Java 2.x API Reference.

## DeleteTargetGroup

Il seguente esempio di codice mostra come utilizzare `DeleteTargetGroup`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
// Deletes the target group.
public void deleteTargetGroup(String targetGroupName) {
    try {
        DescribeTargetGroupsResponse res = getLoadBalancerClient()
            .describeTargetGroups(describe ->
describe.names(targetGroupName));
        getLoadBalancerClient()
            .deleteTargetGroup(builder ->
builder.targetGroupArn(res.targetGroups().get(0).targetGroupArn()));
    } catch (ElasticLoadBalancingV2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }
    System.out.println(targetGroupName + " was deleted.");
}
```

- Per i dettagli sull'API, [DeleteTargetGroup](#) consulta AWS SDK for Java 2.x API Reference.

## DescribeTargetHealth

Il seguente esempio di codice mostra come utilizzare `DescribeTargetHealth`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
// Checks the health of the instances in the target group.
public List<TargetHealthDescription> checkTargetHealth(String targetGroupName) {
    DescribeTargetGroupsRequest targetGroupsRequest =
DescribeTargetGroupsRequest.builder()
        .names(targetGroupName)
        .build();

    DescribeTargetGroupsResponse tgResponse =
getLoadBalancerClient().describeTargetGroups(targetGroupsRequest);
}
```

```
DescribeTargetHealthRequest healthRequest =
DescribeTargetHealthRequest.builder()
    .targetGroupArn(tgResponse.targetGroups().get(0).targetGroupArn())
    .build();

DescribeTargetHealthResponse healthResponse =
getLoadBalancerClient().describeTargetHealth(healthRequest);
return healthResponse.targetHealthDescriptions();
}
```

- Per i dettagli sull'API, [DescribeTargetHealth](#) consulta AWS SDK for Java 2.x API Reference.

## Scenari

### Creazione e gestione di un servizio resiliente

Il seguente esempio di codice mostra come creare un servizio Web con bilanciamento del carico che restituisca consigli su libri, film e canzoni. L'esempio mostra come il servizio risponde ai guasti e spiega come ristrutturarlo per una maggiore resilienza in caso di guasti.

- Utilizza un gruppo con dimensionamento automatico Amazon EC2 per creare istanze Amazon Elastic Compute Cloud (Amazon EC2) basate su un modello di avvio e per mantenere il numero di istanze entro un intervallo specificato.
- Gestisci e distribuisce le richieste HTTP con Elastic Load Balancing.
- Monitora lo stato delle istanze in un gruppo con dimensionamento automatico e inoltra le richieste soltanto alle istanze integre.
- Esegui un server Web Python su ogni istanza EC2 per gestire le richieste HTTP. Il server Web risponde con consigli e controlli dell'integrità.
- Simula un servizio di raccomandazione con una tabella Amazon DynamoDB.
- Controlla la risposta del server web alle richieste e ai controlli sanitari aggiornando AWS Systems Manager i parametri.

## SDK per Java 2.x

 Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Esegui lo scenario interattivo al prompt dei comandi.

```
public class Main {

    public static final String fileName = "C:\\AWS\\resworkflow\\
\\recommendations.json"; // Modify file location.
    public static final String tableName = "doc-example-recommendation-service";
    public static final String startScript = "C:\\AWS\\resworkflow\\
\\server_startup_script.sh"; // Modify file location.
    public static final String policyFile = "C:\\AWS\\resworkflow\\
\\instance_policy.json"; // Modify file location.
    public static final String ssmJSON = "C:\\AWS\\resworkflow\\
\\ssm_only_policy.json"; // Modify file location.
    public static final String failureResponse = "doc-example-resilient-
architecture-failure-response";
    public static final String healthCheck = "doc-example-resilient-architecture-
health-check";
    public static final String templateName = "doc-example-resilience-template";
    public static final String roleName = "doc-example-resilience-role";
    public static final String policyName = "doc-example-resilience-pol";
    public static final String profileName = "doc-example-resilience-prof";

    public static final String badCredsProfileName = "doc-example-resilience-prof-
bc";

    public static final String targetGroupName = "doc-example-resilience-tg";
    public static final String autoScalingGroupName = "doc-example-resilience-
group";
    public static final String lbName = "doc-example-resilience-lb";
    public static final String protocol = "HTTP";
    public static final int port = 80;

    public static final String DASHES = new String(new char[80]).replace("\\0", "-");
```

```
public static void main(String[] args) throws IOException, InterruptedException
{
    Scanner in = new Scanner(System.in);
    Database database = new Database();
    AutoScaler autoScaler = new AutoScaler();
    LoadBalancer loadBalancer = new LoadBalancer();

    System.out.println(DASHES);
    System.out.println("Welcome to the demonstration of How to Build and Manage
a Resilient Service!");
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("A - SETUP THE RESOURCES");
    System.out.println("Press Enter when you're ready to start deploying
resources.");
    in.nextLine();
    deploy(loadBalancer);
    System.out.println(DASHES);
    System.out.println(DASHES);
    System.out.println("B - DEMO THE RESILIENCE FUNCTIONALITY");
    System.out.println("Press Enter when you're ready.");
    in.nextLine();
    demo(loadBalancer);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("C - DELETE THE RESOURCES");
    System.out.println("""
        This concludes the demo of how to build and manage a resilient
service.
        To keep things tidy and to avoid unwanted charges on your account,
we can clean up all AWS resources
        that were created for this demo.
        """);

    System.out.println("\n Do you want to delete the resources (y/n)? ");
    String userInput = in.nextLine().trim().toLowerCase(); // Capture user input

    if (userInput.equals("y")) {
        // Delete resources here
        deleteResources(loadBalancer, autoScaler, database);
        System.out.println("Resources deleted.");
    } else {
```

```

        System.out.println("""
            Okay, we'll leave the resources intact.
            Don't forget to delete them when you're done with them or you
might incur unexpected charges.
            """);
    }
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("The example has completed. ");
    System.out.println("\n Thanks for watching!");
    System.out.println(DASHES);
}

// Deletes the AWS resources used in this example.
private static void deleteResources(LoadBalancer loadBalancer, AutoScaler
autoScaler, Database database)
    throws IOException, InterruptedException {
    loadBalancer.deleteLoadBalancer(lbName);
    System.out.println("*** Wait 30 secs for resource to be deleted");
    TimeUnit.SECONDS.sleep(30);
    loadBalancer.deleteTargetGroup(targetGroupName);
    autoScaler.deleteAutoScaleGroup(autoScalingGroupName);
    autoScaler.deleteRolesPolicies(policyName, roleName, profileName);
    autoScaler.deleteTemplate(templateName);
    database.deleteTable(tableName);
}

private static void deploy(LoadBalancer loadBalancer) throws
InterruptedException, IOException {
    Scanner in = new Scanner(System.in);
    System.out.println(
        """

            For this demo, we'll use the AWS SDK for Java (v2) to create
several AWS resources

            to set up a load-balanced web service endpoint and explore
some ways to make it resilient

            against various kinds of failures.

            Some of the resources create by this demo are:
            \t* A DynamoDB table that the web service depends on to
provide book, movie, and song recommendations.
            \t* An EC2 launch template that defines EC2 instances that
each contain a Python web server.

```

```

        \t* An EC2 Auto Scaling group that manages EC2 instances
across several Availability Zones.
        \t* An Elastic Load Balancing (ELB) load balancer that
targets the Auto Scaling group to distribute requests.
        """);

    System.out.println("Press Enter when you're ready.");
    in.nextLine();
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("Creating and populating a DynamoDB table named " +
tableName);
    Database database = new Database();
    database.createTable(tableName, fileName);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("""
        Creating an EC2 launch template that runs '{startup_script}' when an
instance starts.
        This script starts a Python web server defined in the `server.py`
script. The web server
        listens to HTTP requests on port 80 and responds to requests to '/'
and to '/healthcheck'.
        For demo purposes, this server is run as the root user. In
production, the best practice is to
        run a web server, such as Apache, with least-privileged credentials.

        The template also defines an IAM policy that each instance uses to
assume a role that grants
        permissions to access the DynamoDB recommendation table and Systems
Manager parameters
        that control the flow of the demo.
        """);

    LaunchTemplateCreator templateCreator = new LaunchTemplateCreator();
    templateCreator.createTemplate(policyFile, policyName, profileName,
startScript, templateName, roleName);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println(

```

```
        "Creating an EC2 Auto Scaling group that maintains three EC2
instances, each in a different Availability Zone.");
        System.out.println("*** Wait 30 secs for the VPC to be created");
        TimeUnit.SECONDS.sleep(30);
        AutoScaler autoScaler = new AutoScaler();
        String[] zones = autoScaler.createGroup(3, templateName,
autoScalingGroupName);

        System.out.println("""
            At this point, you have EC2 instances created. Once each instance
starts, it listens for
            HTTP requests. You can see these instances in the console or
continue with the demo.
            Press Enter when you're ready to continue.
            """);

        in.nextLine();
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Creating variables that control the flow of the demo.");
        ParameterHelper paramHelper = new ParameterHelper();
        paramHelper.reset();
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("""
            Creating an Elastic Load Balancing target group and load balancer.
The target group
            defines how the load balancer connects to instances. The load
balancer provides a
            single endpoint where clients connect and dispatches requests to
instances in the group.
            """);

        String vpcId = autoScaler.getDefaultVPC();
        List<Subnet> subnets = autoScaler.getSubnets(vpcId, zones);
        System.out.println("You have retrieved a list with " + subnets.size() + "
subnets");
        String targetGroupArn = loadBalancer.createTargetGroup(protocol, port,
vpcId, targetGroupName);
        String elbDnsName = loadBalancer.createLoadBalancer(subnets, targetGroupArn,
lbName, port, protocol);
```



```

        autoScaler.attachLoadBalancerTargetGroup(autoScalingGroupName,
targetGroupArn);
        System.out.println("Verifying access to the load balancer endpoint...");
        boolean wasSuccessful = loadBalancer.verifyLoadBalancerEndpoint(elbDnsName);
        if (!wasSuccessful) {
            System.out.println("Couldn't connect to the load balancer, verifying
that the port is open...");
            CloseableHttpClient httpClient = HttpClients.createDefault();

            // Create an HTTP GET request to "http://checkip.amazonaws.com"
            HttpGet httpGet = new HttpGet("http://checkip.amazonaws.com");
            try {
                // Execute the request and get the response
                HttpResponse response = httpClient.execute(httpGet);

                // Read the response content.
                String ipAddress =
IOUtils.toString(response.getEntity().getContent(), StandardCharsets.UTF_8).trim();

                // Print the public IP address.
                System.out.println("Public IP Address: " + ipAddress);
                GroupInfo groupInfo = autoScaler.verifyInboundPort(vpcId, port,
ipAddress);
                if (!groupInfo.isPortOpen()) {
                    System.out.println("""
                        For this example to work, the default security group for
your default VPC must
                        allow access from this computer. You can either add it
automatically from this
                        example or add it yourself using the AWS Management
Console.
                        """);

                    System.out.println(
                        "Do you want to add a rule to security group " +
groupInfo.getGroupName() + " to allow");
                    System.out.println("inbound traffic on port " + port + " from
your computer's IP address (y/n) ");
                    String ans = in.nextLine();
                    if ("y".equalsIgnoreCase(ans)) {
                        autoScaler.openInboundPort(groupInfo.getGroupName(),
String.valueOf(port), ipAddress);
                        System.out.println("Security group rule added.");
                    } else {

```

```

        System.out.println("No security group rule added.");
    }
}

} catch (AutoScalingException e) {
    e.printStackTrace();
}
} else if (wasSuccessful) {
    System.out.println("Your load balancer is ready. You can access it by
browsing to:");
    System.out.println("\t http://" + elbDnsName);
} else {
    System.out.println("Couldn't get a successful response from the load
balancer endpoint. Troubleshoot by");
    System.out.println("manually verifying that your VPC and security group
are configured correctly and that");
    System.out.println("you can successfully make a GET request to the load
balancer.");
}

    System.out.println("Press Enter when you're ready to continue with the
demo.");
    in.nextLine();
}

// A method that controls the demo part of the Java program.
public static void demo(LoadBalancer loadBalancer) throws IOException,
InterruptedException {
    ParameterHelper paramHelper = new ParameterHelper();
    System.out.println("Read the ssm_only_policy.json file");
    String ssmOnlyPolicy = readFileAsString(ssmJSON);

    System.out.println("Resetting parameters to starting values for demo.");
    paramHelper.reset();

    System.out.println(
        """"
                This part of the demonstration shows how to toggle
different parts of the system
                to create situations where the web service fails, and shows
how using a resilient
                architecture can keep the web service running in spite of
these failures.
        """);
}

```

```
        At the start, the load balancer endpoint returns
recommendations and reports that all targets are healthy.
        """);
        demoChoices(loadBalancer);

        System.out.println(
            ""
                The web service running on the EC2 instances gets
recommendations by querying a DynamoDB table.
                The table name is contained in a Systems Manager parameter
named self.param_helper.table.
                To simulate a failure of the recommendation service, let's
set this parameter to name a non-existent table.
            """);
        paramHelper.put(paramHelper.tableName, "this-is-not-a-table");

        System.out.println(
            ""
                \nNow, sending a GET request to the load balancer endpoint
returns a failure code. But, the service reports as
                healthy to the load balancer because shallow health checks
don't check for failure of the recommendation service.
            """);
        demoChoices(loadBalancer);

        System.out.println(
            ""
                Instead of failing when the recommendation service fails,
the web service can return a static response.
                While this is not a perfect solution, it presents the
customer with a somewhat better experience than failure.
            """);
        paramHelper.put(paramHelper.failureResponse, "static");

        System.out.println("""
            Now, sending a GET request to the load balancer endpoint returns a
static response.
            The service still reports as healthy because health checks are still
shallow.
            """);
        demoChoices(loadBalancer);

        System.out.println("Let's reinstate the recommendation service.");
        paramHelper.put(paramHelper.tableName, paramHelper.dyntable);
```

```
System.out.println("""
    Let's also substitute bad credentials for one of the instances in
the target group so that it can't
    access the DynamoDB recommendation table. We will get an instance id
value.
    """);

LaunchTemplateCreator templateCreator = new LaunchTemplateCreator();
AutoScaler autoScaler = new AutoScaler();

// Create a new instance profile based on badCredsProfileName.
templateCreator.createInstanceProfile(policyFile, policyName,
badCredsProfileName, roleName);
String badInstanceId = autoScaler.getBadInstance(autoScalingGroupName);
System.out.println("The bad instance id values used for this demo is " +
badInstanceId);

String profileAssociationId = autoScaler.getInstanceProfile(badInstanceId);
System.out.println("The association Id value is " + profileAssociationId);
System.out.println("Replacing the profile for instance " + badInstanceId
    + " with a profile that contains bad credentials");
autoScaler.replaceInstanceProfile(badInstanceId, badCredsProfileName,
profileAssociationId);

System.out.println(
    """
        Now, sending a GET request to the load balancer endpoint
returns either a recommendation or a static response,
        depending on which instance is selected by the load
balancer.
    """);

demoChoices(loadBalancer);

System.out.println("""
    Let's implement a deep health check. For this demo, a deep health
check tests whether
    the web service can access the DynamoDB table that it depends on for
recommendations. Note that
    the deep health check is only for ELB routing and not for Auto
Scaling instance health.
    This kind of deep health check is not recommended for Auto Scaling
instance health, because it
```

```
        risks accidental termination of all instances in the Auto Scaling
group when a dependent service fails.
        """);

        System.out.println("""
        By implementing deep health checks, the load balancer can detect
when one of the instances is failing
        and take that instance out of rotation.
        """);

        paramHelper.put(paramHelper.healthCheck, "deep");

        System.out.println("""
        Now, checking target health indicates that the instance with bad
credentials
        is unhealthy. Note that it might take a minute or two for the load
balancer to detect the unhealthy
        instance. Sending a GET request to the load balancer endpoint always
returns a recommendation, because
        the load balancer takes unhealthy instances out of its rotation.
        """);

        demoChoices(loadBalancer);

        System.out.println(
        ""
                "        Because the instances in this demo are controlled by an auto
scaler, the simplest way to fix an unhealthy
                instance is to terminate it and let the auto scaler start a
new instance to replace it.
                """);
        autoScaler.terminateInstance(badInstanceId);

        System.out.println("""
        Even while the instance is terminating and the new instance is
starting, sending a GET
        request to the web service continues to get a successful
recommendation response because
        the load balancer routes requests to the healthy instances. After
the replacement instance
        starts and reports as healthy, it is included in the load balancing
rotation.

        Note that terminating and replacing an instance typically takes
several minutes, during which time you
```

```

        can see the changing health check status until the new instance is
running and healthy.
        """);

        demoChoices(loadBalancer);
        System.out.println(
            "If the recommendation service fails now, deep health checks mean
all instances report as unhealthy.");
        paramHelper.put(paramHelper.tableName, "this-is-not-a-table");

        demoChoices(loadBalancer);
        paramHelper.reset();
    }

    public static void demoChoices(LoadBalancer loadBalancer) throws IOException,
InterruptedException {
        String[] actions = {
            "Send a GET request to the load balancer endpoint.",
            "Check the health of load balancer targets.",
            "Go to the next part of the demo."
        };
        Scanner scanner = new Scanner(System.in);

        while (true) {
            System.out.println("-".repeat(88));
            System.out.println("See the current state of the service by selecting
one of the following choices:");
            for (int i = 0; i < actions.length; i++) {
                System.out.println(i + ": " + actions[i]);
            }

            try {
                System.out.print("\nWhich action would you like to take? ");
                int choice = scanner.nextInt();
                System.out.println("-".repeat(88));

                switch (choice) {
                    case 0 -> {
                        System.out.println("Request:\n");
                        System.out.println("GET http://" +
loadBalancer.getEndpoint(lbName));
                        CloseableHttpClient httpClient =
HttpClients.createDefault();

```

```

        // Create an HTTP GET request to the ELB.
        HttpGet httpGet = new HttpGet("http://" +
loadBalancer.getEndpoint(lbName));

        // Execute the request and get the response.
        HttpResponse response = httpClient.execute(httpGet);
        int statusCode = response.getStatusLine().getStatusCode();
        System.out.println("HTTP Status Code: " + statusCode);

        // Display the JSON response
        BufferedReader reader = new BufferedReader(
            new
InputStreamReader(response.getEntity().getContent()));
        StringBuilder jsonResponse = new StringBuilder();
        String line;
        while ((line = reader.readLine()) != null) {
            jsonResponse.append(line);
        }
        reader.close();

        // Print the formatted JSON response.
        System.out.println("Full Response:\n");
        System.out.println(jsonResponse.toString());

        // Close the HTTP client.
        httpClient.close();
    }
    case 1 -> {
        System.out.println("\nChecking the health of load balancer
targets:\n");

        List<TargetHealthDescription> health =
loadBalancer.checkTargetHealth(targetGroupName);
        for (TargetHealthDescription target : health) {
            System.out.printf("\tTarget %s on port %d is %s\n",
target.target().id(),
                target.target().port(),
target.targetHealth().stateAsString());
        }
        System.out.println("""
            Note that it can take a minute or two for the health
check to update

            after changes are made.
            """);
    }
}

```

```

        }
        case 2 -> {
            System.out.println("\n0kay, let's move on.");
            System.out.println("-".repeat(88));
            return; // Exit the method when choice is 2
        }
        default -> System.out.println("You must choose a value between
0-2. Please select again.");
    }

    } catch (java.util.InputMismatchException e) {
        System.out.println("Invalid input. Please select again.");
        scanner.nextLine(); // Clear the input buffer.
    }
}

}

public static String readFileAsString(String filePath) throws IOException {
    byte[] bytes = Files.readAllBytes(Paths.get(filePath));
    return new String(bytes);
}
}

```

Crea una classe che racchiuda le operazioni di dimensionamento automatico e Amazon EC2.

```

public class AutoScaler {

    private static Ec2Client ec2Client;
    private static AutoScalingClient autoScalingClient;
    private static IamClient iamClient;

    private static SsmClient ssmClient;

    private IamClient getIAMClient() {
        if (iamClient == null) {
            iamClient = IamClient.builder()
                .region(Region.US_EAST_1)
                .build();
        }
        return iamClient;
    }
}

```



```
private SsmClient getSSMClient() {
    if (ssmClient == null) {
        ssmClient = SsmClient.builder()
            .region(Region.US_EAST_1)
            .build();
    }
    return ssmClient;
}

private Ec2Client getEc2Client() {
    if (ec2Client == null) {
        ec2Client = Ec2Client.builder()
            .region(Region.US_EAST_1)
            .build();
    }
    return ec2Client;
}

private AutoScalingClient getAutoScalingClient() {
    if (autoScalingClient == null) {
        autoScalingClient = AutoScalingClient.builder()
            .region(Region.US_EAST_1)
            .build();
    }
    return autoScalingClient;
}

/**
 * Terminates and instances in an EC2 Auto Scaling group. After an instance is
 * terminated, it can no longer be accessed.
 */
public void terminateInstance(String instanceId) {
    TerminateInstanceInAutoScalingGroupRequest terminateInstanceIRequest =
    TerminateInstanceInAutoScalingGroupRequest
        .builder()
        .instanceId(instanceId)
        .shouldDecrementDesiredCapacity(false)
        .build();

    getAutoScalingClient().terminateInstanceInAutoScalingGroup(terminateInstanceIRequest);
    System.out.format("Terminated instance %s.", instanceId);
}
}
```

```
/**
 * Replaces the profile associated with a running instance. After the profile is
 * replaced, the instance is rebooted to ensure that it uses the new profile.
 * When
 * the instance is ready, Systems Manager is used to restart the Python web
 * server.
 */
public void replaceInstanceProfile(String instanceId, String
newInstanceProfileName, String profileAssociationId)
    throws InterruptedException {
    // Create an IAM instance profile specification.
    software.amazon.awssdk.services.ec2.model.IamInstanceProfileSpecification
iamInstanceProfile =
software.amazon.awssdk.services.ec2.model.IamInstanceProfileSpecification
    .builder()
    .name(newInstanceProfileName) // Make sure 'newInstanceProfileName'
is a valid IAM Instance Profile
    .build();

    // Replace the IAM instance profile association for the EC2 instance.
    ReplaceIamInstanceProfileAssociationRequest replaceRequest =
ReplaceIamInstanceProfileAssociationRequest
    .builder()
    .iamInstanceProfile(iamInstanceProfile)
    .associationId(profileAssociationId) // Make sure
'profileAssociationId' is a valid association ID.
    .build();

    try {
        getEc2Client().replaceIamInstanceProfileAssociation(replaceRequest);
        // Handle the response as needed.
    } catch (Ec2Exception e) {
        // Handle exceptions, log, or report the error.
        System.err.println("Error: " + e.getMessage());
    }

    System.out.format("Replaced instance profile for association %s with profile
%s.", profileAssociationId,
        newInstanceProfileName);
    TimeUnit.SECONDS.sleep(15);
    boolean instReady = false;
    int tries = 0;

    // Reboot after 60 seconds
```

```
        while (!instReady) {
            if (tries % 6 == 0) {
                getEc2Client().rebootInstances(RebootInstancesRequest.builder()
                    .instanceIds(instanceId)
                    .build());
                System.out.println("Rebooting instance " + instanceId + " and
waiting for it to be ready.");
            }
            tries++;
            try {
                TimeUnit.SECONDS.sleep(10);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }

            DescribeInstanceInformationResponse informationResponse =
getSSMClient().describeInstanceInformation();
            List<InstanceInformation> instanceInformationList =
informationResponse.instanceInformationList();
            for (InstanceInformation info : instanceInformationList) {
                if (info.instanceId().equals(instanceId)) {
                    instReady = true;
                    break;
                }
            }
        }

        SendCommandRequest sendCommandRequest = SendCommandRequest.builder()
            .instanceIds(instanceId)
            .documentName("AWS-RunShellScript")
            .parameters(Collections.singletonMap("commands",
                Collections.singletonList("cd / && sudo python3 server.py
80")))
            .build();

        getSSMClient().sendCommand(sendCommandRequest);
        System.out.println("Restarted the Python web server on instance " +
instanceId + ".");
    }

    public void openInboundPort(String secGroupId, String port, String ipAddress) {
        AuthorizeSecurityGroupIngressRequest ingressRequest =
AuthorizeSecurityGroupIngressRequest.builder()
            .groupName(secGroupId)
```

```

        .cidrIp(ipAddress)
        .fromPort(Integer.parseInt(port))
        .build();

    getEc2Client().authorizeSecurityGroupIngress(ingressRequest);
    System.out.format("Authorized ingress to %s on port %s from %s.",
secGroupId, port, ipAddress);
}

/**
 * Detaches a role from an instance profile, detaches policies from the role,
 * and deletes all the resources.
 */
public void deleteInstanceProfile(String roleName, String profileName) {
    try {
        software.amazon.awssdk.services.iam.model.GetInstanceProfileRequest
getInstanceProfileRequest =
software.amazon.awssdk.services.iam.model.GetInstanceProfileRequest
        .builder()
        .instanceProfileName(profileName)
        .build();

        GetInstanceProfileResponse response =
getIAMClient().getInstanceProfile(getInstanceProfileRequest);
        String name = response.getInstanceProfile().getInstanceProfileName();
        System.out.println(name);

        RemoveRoleFromInstanceProfileRequest profileRequest =
RemoveRoleFromInstanceProfileRequest.builder()
        .instanceProfileName(profileName)
        .roleName(roleName)
        .build();

        getIAMClient().removeRoleFromInstanceProfile(profileRequest);
        DeleteInstanceProfileRequest deleteInstanceProfileRequest =
DeleteInstanceProfileRequest.builder()
        .instanceProfileName(profileName)
        .build();

        getIAMClient().deleteInstanceProfile(deleteInstanceProfileRequest);
        System.out.println("Deleted instance profile " + profileName);

        DeleteRoleRequest deleteRoleRequest = DeleteRoleRequest.builder()
        .roleName(roleName)

```

```

        .build();

        // List attached role policies.
        ListAttachedRolePoliciesResponse rolesResponse = getIAMClient()
            .listAttachedRolePolicies(role -> role.roleName(roleName));
        List<AttachedPolicy> attachedPolicies =
rolesResponse.attachedPolicies();
        for (AttachedPolicy attachedPolicy : attachedPolicies) {
            DetachRolePolicyRequest request = DetachRolePolicyRequest.builder()
                .roleName(roleName)
                .policyArn(attachedPolicy.policyArn())
                .build();

            getIAMClient().detachRolePolicy(request);
            System.out.println("Detached and deleted policy " +
attachedPolicy.policyName());
        }

        getIAMClient().deleteRole(deleteRoleRequest);
        System.out.println("Instance profile and role deleted.");

    } catch (IamException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public void deleteTemplate(String templateName) {
    getEc2Client().deleteLaunchTemplate(name ->
name.launchTemplateName(templateName));
    System.out.format(templateName + " was deleted.");
}

public void deleteAutoScaleGroup(String groupName) {
    DeleteAutoScalingGroupRequest deleteAutoScalingGroupRequest =
DeleteAutoScalingGroupRequest.builder()
        .autoScalingGroupName(groupName)
        .forceDelete(true)
        .build();

    getAutoScalingClient().deleteAutoScalingGroup(deleteAutoScalingGroupRequest);
    System.out.println(groupName + " was deleted.");
}

```

```
/*
 * Verify the default security group of the specified VPC allows ingress from
 * this
 * computer. This can be done by allowing ingress from this computer's IP
 * address. In some situations, such as connecting from a corporate network, you
 * must instead specify a prefix list ID. You can also temporarily open the port
 * to
 * any IP address while running this example. If you do, be sure to remove
 * public
 * access when you're done.
 *
 */
public GroupInfo verifyInboundPort(String VPC, int port, String ipAddress) {
    boolean portIsOpen = false;
    GroupInfo groupInfo = new GroupInfo();
    try {
        Filter filter = Filter.builder()
            .name("group-name")
            .values("default")
            .build();

        Filter filter1 = Filter.builder()
            .name("vpc-id")
            .values(VPC)
            .build();

        DescribeSecurityGroupsRequest securityGroupsRequest =
DescribeSecurityGroupsRequest.builder()
            .filters(filter, filter1)
            .build();

        DescribeSecurityGroupsResponse securityGroupsResponse = getEc2Client()
            .describeSecurityGroups(securityGroupsRequest);
        String securityGroup =
securityGroupsResponse.securityGroups().get(0).groupName();
        groupInfo.setGroupName(securityGroup);

        for (SecurityGroup secGroup : securityGroupsResponse.securityGroups()) {
            System.out.println("Found security group: " + secGroup.groupId());

            for (IpPermission ipPermission : secGroup.ipPermissions()) {
                if (ipPermission.fromPort() == port) {
                    System.out.println("Found inbound rule: " + ipPermission);
                }
            }
        }
    }
}
```

```

        for (IpRange ipRange : ipPermission.ipRanges()) {
            String cidrIp = ipRange.cidrIp();
            if (cidrIp.startsWith(ipAddress) ||
cidrIp.equals("0.0.0.0/0")) {
                System.out.println(cidrIp + " is applicable");
                portIsOpen = true;
            }
        }

        if (!ipPermission.prefixListIds().isEmpty()) {
            System.out.println("Prefix lList is applicable");
            portIsOpen = true;
        }

        if (!portIsOpen) {
            System.out
                .println("The inbound rule does not appear to be
open to either this computer's IP,"
                        + " all IP addresses (0.0.0.0/0), or to
a prefix list ID.");
        } else {
            break;
        }
    }
}

} catch (AutoScalingException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
}

groupInfo.setPortOpen(portIsOpen);
return groupInfo;
}

/**
 * Attaches an Elastic Load Balancing (ELB) target group to this EC2 Auto
 * Scaling group.
 * The target group specifies how the load balancer forward requests to the
 * instances
 * in the group.
 */
public void attachLoadBalancerTargetGroup(String asGroupName, String
targetGroupARN) {

```

```
        try {
            AttachLoadBalancerTargetGroupsRequest targetGroupsRequest =
AttachLoadBalancerTargetGroupsRequest.builder()
                .autoScalingGroupName(asGroupName)
                .targetGroupARNs(targetGroupARN)
                .build();

getAutoScalingClient().attachLoadBalancerTargetGroups(targetGroupsRequest);
            System.out.println("Attached load balancer to " + asGroupName);

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    // Creates an EC2 Auto Scaling group with the specified size.
    public String[] createGroup(int groupSize, String templateName, String
autoScalingGroupName) {

        // Get availability zones.
        software.amazon.awssdk.services.ec2.model.DescribeAvailabilityZonesRequest
zonesRequest =
software.amazon.awssdk.services.ec2.model.DescribeAvailabilityZonesRequest
                .builder()
                .build();

        DescribeAvailabilityZonesResponse zonesResponse =
getEc2Client().describeAvailabilityZones(zonesRequest);
        List<String> availabilityZoneNames =
zonesResponse.availabilityZones().stream()

.map(software.amazon.awssdk.services.ec2.model.AvailabilityZone::zoneName)
                .collect(Collectors.toList());

        String availabilityZones = String.join(",", availabilityZoneNames);
        LaunchTemplateSpecification specification =
LaunchTemplateSpecification.builder()
                .launchTemplateName(templateName)
                .version("$Default")
                .build();

        String[] zones = availabilityZones.split(",");
```



```
        CreateAutoScalingGroupRequest groupRequest =
CreateAutoScalingGroupRequest.builder()
    .launchTemplate(specification)
    .availabilityZones(zones)
    .maxSize(groupSize)
    .minSize(groupSize)
    .autoScalingGroupName(autoScalingGroupName)
    .build();

    try {
        getAutoScalingClient().createAutoScalingGroup(groupRequest);

    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    System.out.println("Created an EC2 Auto Scaling group named " +
autoScalingGroupName);
    return zones;
}

public String getDefaultVPC() {
    // Define the filter.
    Filter defaultFilter = Filter.builder()
        .name("is-default")
        .values("true")
        .build();

    software.amazon.awssdk.services.ec2.model.DescribeVpcsRequest request =
software.amazon.awssdk.services.ec2.model.DescribeVpcsRequest
        .builder()
        .filters(defaultFilter)
        .build();

    DescribeVpcsResponse response = getEc2Client().describeVpcs(request);
    return response.vpcs().get(0).vpcId();
}

// Gets the default subnets in a VPC for a specified list of Availability Zones.
public List<Subnet> getSubnets(String vpcId, String[] availabilityZones) {
    List<Subnet> subnets = null;
    Filter vpcFilter = Filter.builder()
        .name("vpc-id")
        .values(vpcId)
```

```
        .build();

    Filter azFilter = Filter.builder()
        .name("availability-zone")
        .values(availabilityZones)
        .build();

    Filter defaultForAZ = Filter.builder()
        .name("default-for-az")
        .values("true")
        .build();

    DescribeSubnetsRequest request = DescribeSubnetsRequest.builder()
        .filters(vpcFilter, azFilter, defaultForAZ)
        .build();

    DescribeSubnetsResponse response = getEc2Client().describeSubnets(request);
    subnets = response.subnets();
    return subnets;
}

// Gets data about the instances in the EC2 Auto Scaling group.
public String getBadInstance(String groupName) {
    DescribeAutoScalingGroupsRequest request =
DescribeAutoScalingGroupsRequest.builder()
        .autoScalingGroupNames(groupName)
        .build();

    DescribeAutoScalingGroupsResponse response =
getAutoScalingClient().describeAutoScalingGroups(request);
    AutoScalingGroup autoScalingGroup = response.autoScalingGroups().get(0);
    List<String> instanceIds = autoScalingGroup.instances().stream()
        .map(instance -> instance.instanceId())
        .collect(Collectors.toList());

    String[] instanceIdArray = instanceIds.toArray(new String[0]);
    for (String instanceId : instanceIdArray) {
        System.out.println("Instance ID: " + instanceId);
        return instanceId;
    }
    return "";
}

// Gets data about the profile associated with an instance.
```

```

public String getInstanceProfile(String instanceId) {
    Filter filter = Filter.builder()
        .name("instance-id")
        .values(instanceId)
        .build();

    DescribeIamInstanceProfileAssociationsRequest associationsRequest =
DescribeIamInstanceProfileAssociationsRequest
        .builder()
        .filters(filter)
        .build();

    DescribeIamInstanceProfileAssociationsResponse response = getEc2Client()
        .describeIamInstanceProfileAssociations(associationsRequest);
    return response.iamInstanceProfileAssociations().get(0).associationId();
}

public void deleteRolesPolicies(String policyName, String roleName, String
InstanceProfile) {
    ListPoliciesRequest listPoliciesRequest =
ListPoliciesRequest.builder().build();
    ListPoliciesResponse listPoliciesResponse =
getIAMClient().listPolicies(listPoliciesRequest);
    for (Policy policy : listPoliciesResponse.policies()) {
        if (policy.policyName().equals(policyName)) {
            // List the entities (users, groups, roles) that are attached to the
policy.

software.amazon.awssdk.services.iam.model.ListEntitiesForPolicyRequest
listEntitiesRequest =
software.amazon.awssdk.services.iam.model.ListEntitiesForPolicyRequest
        .builder()
        .policyArn(policy.arn())
        .build();
        ListEntitiesForPolicyResponse listEntitiesResponse = iamClient
            .listEntitiesForPolicy(listEntitiesRequest);
        if (!listEntitiesResponse.policyGroups().isEmpty() || !
listEntitiesResponse.policyUsers().isEmpty()
            || !listEntitiesResponse.policyRoles().isEmpty()) {
            // Detach the policy from any entities it is attached to.
            DetachRolePolicyRequest detachPolicyRequest =
DetachRolePolicyRequest.builder()
                .policyArn(policy.arn())
                .roleName(roleName) // Specify the name of the IAM role

```

```
        .build();

        getIAMClient().detachRolePolicy(detachPolicyRequest);
        System.out.println("Policy detached from entities.");
    }

    // Now, you can delete the policy.
    DeletePolicyRequest deletePolicyRequest =
DeletePolicyRequest.builder()
        .policyArn(policy.arn())
        .build();

    getIAMClient().deletePolicy(deletePolicyRequest);
    System.out.println("Policy deleted successfully.");
    break;
}
}

// List the roles associated with the instance profile
ListInstanceProfilesForRoleRequest listRolesRequest =
ListInstanceProfilesForRoleRequest.builder()
    .roleName(roleName)
    .build();

// Detach the roles from the instance profile
ListInstanceProfilesForRoleResponse listRolesResponse =
iamClient.listInstanceProfilesForRole(listRolesRequest);
for (software.amazon.awssdk.services.iam.model.InstanceProfile profile :
listRolesResponse.instanceProfiles()) {
    RemoveRoleFromInstanceProfileRequest removeRoleRequest =
RemoveRoleFromInstanceProfileRequest.builder()
        .instanceProfileName(InstanceProfile)
        .roleName(roleName) // Remove the extra dot here
        .build();

    getIAMClient().removeRoleFromInstanceProfile(removeRoleRequest);
    System.out.println("Role " + roleName + " removed from instance profile
" + InstanceProfile);
}

// Delete the instance profile after removing all roles
DeleteInstanceProfileRequest deleteInstanceProfileRequest =
DeleteInstanceProfileRequest.builder()
    .instanceProfileName(InstanceProfile)
```

```

        .build();

        getIAMClient().deleteInstanceProfile(r ->
r.instanceProfileName(InstanceProfile));
        System.out.println(InstanceProfile + " Deleted");
        System.out.println("All roles and policies are deleted.");
    }
}

```

Crea una classe che racchiuda le operazioni di Elastic Load Balancing.

```

public class LoadBalancer {
    public ElasticLoadBalancingV2Client elasticLoadBalancingV2Client;

    public ElasticLoadBalancingV2Client getLoadBalancerClient() {
        if (elasticLoadBalancingV2Client == null) {
            elasticLoadBalancingV2Client = ElasticLoadBalancingV2Client.builder()
                .region(Region.US_EAST_1)
                .build();
        }

        return elasticLoadBalancingV2Client;
    }

    // Checks the health of the instances in the target group.
    public List<TargetHealthDescription> checkTargetHealth(String targetGroupName) {
        DescribeTargetGroupsRequest targetGroupsRequest =
DescribeTargetGroupsRequest.builder()
            .names(targetGroupName)
            .build();

        DescribeTargetGroupsResponse tgResponse =
getLoadBalancerClient().describeTargetGroups(targetGroupsRequest);

        DescribeTargetHealthRequest healthRequest =
DescribeTargetHealthRequest.builder()
            .targetGroupArn(tgResponse.targetGroups().get(0).targetGroupArn())
            .build();

        DescribeTargetHealthResponse healthResponse =
getLoadBalancerClient().describeTargetHealth(healthRequest);
        return healthResponse.targetHealthDescriptions();
    }
}

```

```
}

// Gets the HTTP endpoint of the load balancer.
public String getEndpoint(String lbName) {
    DescribeLoadBalancersResponse res = getLoadBalancerClient()
        .describeLoadBalancers(describe -> describe.names(lbName));
    return res.loadBalancers().get(0).dnsName();
}

// Deletes a load balancer.
public void deleteLoadBalancer(String lbName) {
    try {
        // Use a waiter to delete the Load Balancer.
        DescribeLoadBalancersResponse res = getLoadBalancerClient()
            .describeLoadBalancers(describe -> describe.names(lbName));
        ElasticLoadBalancingV2Waiter loadBalancerWaiter =
getLoadBalancerClient().waiter();
        DescribeLoadBalancersRequest request =
DescribeLoadBalancersRequest.builder()
            .loadBalancerArns(res.loadBalancers().get(0).loadBalancerArn())
            .build();

        getLoadBalancerClient().deleteLoadBalancer(
            builder ->
builder.loadBalancerArn(res.loadBalancers().get(0).loadBalancerArn()));
        WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =
loadBalancerWaiter
            .waitUntilLoadBalancersDeleted(request);
        waiterResponse.matched().response().ifPresent(System.out::println);

    } catch (ElasticLoadBalancingV2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }
    System.out.println(lbName + " was deleted.");
}

// Deletes the target group.
public void deleteTargetGroup(String targetGroupName) {
    try {
        DescribeTargetGroupsResponse res = getLoadBalancerClient()
            .describeTargetGroups(describe ->
describe.names(targetGroupName));
        getLoadBalancerClient()
```

```

        .deleteTargetGroup(builder ->
builder.targetGroupArn(res.targetGroups().get(0).targetGroupArn()));
    } catch (ElasticLoadBalancingV2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }
    System.out.println(targetGroupName + " was deleted.");
}

// Verify this computer can successfully send a GET request to the load balancer
// endpoint.
public boolean verifyLoadBalancerEndpoint(String elbDnsName) throws IOException,
InterruptedException {
    boolean success = false;
    int retries = 3;
    CloseableHttpClient httpClient = HttpClients.createDefault();

    // Create an HTTP GET request to the ELB.
    HttpGet httpGet = new HttpGet("http://" + elbDnsName);
    try {
        while ((!success) && (retries > 0)) {
            // Execute the request and get the response.
            HttpResponse response = httpClient.execute(httpGet);
            int statusCode = response.getStatusLine().getStatusCode();
            System.out.println("HTTP Status Code: " + statusCode);
            if (statusCode == 200) {
                success = true;
            } else {
                retries--;
                System.out.println("Got connection error from load balancer
endpoint, retrying...");
                TimeUnit.SECONDS.sleep(15);
            }
        }

    } catch (org.apache.http.conn.HttpHostConnectException e) {
        System.out.println(e.getMessage());
    }

    System.out.println("Status.." + success);
    return success;
}

/*
 * Creates an Elastic Load Balancing target group. The target group specifies

```

```
    * how
    * the load balancer forward requests to instances in the group and how instance
    * health is checked.
    */
    public String createTargetGroup(String protocol, int port, String vpcId, String
targetGroupName) {
        CreateTargetGroupRequest targetGroupRequest =
CreateTargetGroupRequest.builder()
            .healthCheckPath("/healthcheck")
            .healthCheckTimeoutSeconds(5)
            .port(port)
            .vpcId(vpcId)
            .name(targetGroupName)
            .protocol(protocol)
            .build();

        CreateTargetGroupResponse targetGroupResponse =
getLoadBalancerClient().createTargetGroup(targetGroupRequest);
        String targetGroupArn =
targetGroupResponse.targetGroups().get(0).targetGroupArn();
        String targetGroup =
targetGroupResponse.targetGroups().get(0).targetGroupName();
        System.out.println("The " + targetGroup + " was created with ARN" +
targetGroupArn);
        return targetGroupArn;
    }

    /**
     * Creates an Elastic Load Balancing load balancer that uses the specified
     * subnets
     * and forwards requests to the specified target group.
     */
    public String createLoadBalancer(List<Subnet> subnetIds, String targetGroupARN,
String lbName, int port,
        String protocol) {
        try {
            List<String> subnetIdStrings = subnetIds.stream()
                .map(Subnet::subnetId)
                .collect(Collectors.toList());

            CreateLoadBalancerRequest balancerRequest =
CreateLoadBalancerRequest.builder()
                .subnets(subnetIdStrings)
                .name(lbName)
```



```
        .scheme("internet-facing")
        .build();

    // Create and wait for the load balancer to become available.
    CreateLoadBalancerResponse lsResponse =
getLoadBalancerClient().createLoadBalancer(balancerRequest);
    String lbARN = lsResponse.loadBalancers().get(0).loadBalancerArn();

    ElasticLoadBalancingV2Waiter loadBalancerWaiter =
getLoadBalancerClient().waiter();
    DescribeLoadBalancersRequest request =
DescribeLoadBalancersRequest.builder()
        .loadBalancerArns(lbARN)
        .build();

    System.out.println("Waiting for Load Balancer " + lbName + " to become
available.");
    WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =
loadBalancerWaiter
        .waitUntilLoadBalancerAvailable(request);
    waiterResponse.matched().response().ifPresent(System.out::println);
    System.out.println("Load Balancer " + lbName + " is available.");

    // Get the DNS name (endpoint) of the load balancer.
    String lbDNSName = lsResponse.loadBalancers().get(0).dnsName();
    System.out.println("*** Load Balancer DNS Name: " + lbDNSName);

    // Create a listener for the load balance.
    Action action = Action.builder()
        .targetGroupArn(targetGroupARN)
        .type("forward")
        .build();

    CreateListenerRequest listenerRequest = CreateListenerRequest.builder()

.loadBalancerArn(lsResponse.loadBalancers().get(0).loadBalancerArn())
        .defaultActions(action)
        .port(port)
        .protocol(protocol)
        .defaultActions(action)
        .build();

    getLoadBalancerClient().createListener(listenerRequest);
```

```

        System.out.println("Created listener to forward traffic from load
balancer " + lbName + " to target group "
            + targetGroupARN);

        // Return the load balancer DNS name.
        return lbDNSName;

    } catch (ElasticLoadBalancingV2Exception e) {
        e.printStackTrace();
    }
    return "";
}
}

```

Crea una classe che utilizzi DynamoDB per simulare un servizio di raccomandazione.

```

public class Database {

    private static DynamoDbClient dynamoDbClient;

    public static DynamoDbClient getDynamoDbClient() {
        if (dynamoDbClient == null) {
            dynamoDbClient = DynamoDbClient.builder()
                .region(Region.US_EAST_1)
                .build();
        }
        return dynamoDbClient;
    }

    // Checks to see if the Amazon DynamoDB table exists.
    private boolean doesTableExist(String tableName) {
        try {
            // Describe the table and catch any exceptions.
            DescribeTableRequest describeTableRequest =
DescribeTableRequest.builder()
                .tableName(tableName)
                .build();

            getDynamoDbClient().describeTable(describeTableRequest);
            System.out.println("Table '" + tableName + "' exists.");
            return true;
        }
    }
}

```

```
    } catch (ResourceNotFoundException e) {
        System.out.println("Table '" + tableName + "' does not exist.");
    } catch (DynamoDbException e) {
        System.err.println("Error checking table existence: " + e.getMessage());
    }
    return false;
}

/**
 * Creates a DynamoDB table to use a recommendation service. The table has a
 * hash key named 'MediaType' that defines the type of media recommended, such
 * as
 * Book or Movie, and a range key named 'ItemId' that, combined with the
 * MediaType,
 * forms a unique identifier for the recommended item.
 */
public void createTable(String tableName, String fileName) throws IOException {
    // First check to see if the table exists.
    boolean doesExist = doesTableExist(tableName);
    if (!doesExist) {
        DynamoDbWaiter dbWaiter = getDynamoDbClient().waiter();
        CreateTableRequest createTableRequest = CreateTableRequest.builder()
            .tableName(tableName)
            .attributeDefinitions(
                AttributeDefinition.builder()
                    .attributeName("MediaType")
                    .attributeType(ScalarAttributeType.S)
                    .build(),
                AttributeDefinition.builder()
                    .attributeName("ItemId")
                    .attributeType(ScalarAttributeType.N)
                    .build())
            .keySchema(
                KeySchemaElement.builder()
                    .attributeName("MediaType")
                    .keyType(KeyType.HASH)
                    .build(),
                KeySchemaElement.builder()
                    .attributeName("ItemId")
                    .keyType(KeyType.RANGE)
                    .build())
            .provisionedThroughput(
                ProvisionedThroughput.builder()
                    .readCapacityUnits(5L)
    }
```

```

        .writeCapacityUnits(5L)
        .build())
        .build();

    getDynamoDbClient().createTable(createTableRequest);
    System.out.println("Creating table " + tableName + "...");

    // Wait until the Amazon DynamoDB table is created.
    DescribeTableRequest tableRequest = DescribeTableRequest.builder()
        .tableName(tableName)
        .build();

    WaiterResponse<DescribeTableResponse> waiterResponse =
    dbWaiter.waitUntilTableExists(tableRequest);
    waiterResponse.matched().response().ifPresent(System.out::println);
    System.out.println("Table " + tableName + " created.");

    // Add records to the table.
    populateTable(fileName, tableName);
}

public void deleteTable(String tableName) {
    getDynamoDbClient().deleteTable(table -> table.tableName(tableName));
    System.out.println("Table " + tableName + " deleted.");
}

// Populates the table with data located in a JSON file using the DynamoDB
// enhanced client.
public void populateTable(String fileName, String tableName) throws IOException
{
    DynamoDbEnhancedClient enhancedClient = DynamoDbEnhancedClient.builder()
        .dynamoDbClient(getDynamoDbClient())
        .build();
    ObjectMapper objectMapper = new ObjectMapper();
    File jsonFile = new File(fileName);
    JsonNode rootNode = objectMapper.readTree(jsonFile);

    DynamoDbTable<Recommendation> mappedTable = enhancedClient.table(tableName,
        TableSchema.fromBean(Recommendation.class));
    for (JsonNode currentNode : rootNode) {
        String mediaType = currentNode.path("MediaType").path("S").asText();
        int itemId = currentNode.path("ItemId").path("N").asInt();
        String title = currentNode.path("Title").path("S").asText();
    }
}

```

```

        String creator = currentNode.path("Creator").path("S").asText();

        // Create a Recommendation object and set its properties.
        Recommendation rec = new Recommendation();
        rec.setMediaType(mediaType);
        rec.setItemId(itemId);
        rec.setTitle(title);
        rec.setCreator(creator);

        // Put the item into the DynamoDB table.
        mappedTable.putItem(rec); // Add the Recommendation to the list.
    }
    System.out.println("Added all records to the " + tableName);
}
}

```

Crea una classe che racchiuda le operazioni di Systems Manager.

```

public class ParameterHelper {

    String tableName = "doc-example-resilient-architecture-table";
    String dyntable = "doc-example-recommendation-service";
    String failureResponse = "doc-example-resilient-architecture-failure-response";
    String healthCheck = "doc-example-resilient-architecture-health-check";

    public void reset() {
        put(dyntable, tableName);
        put(failureResponse, "none");
        put(healthCheck, "shallow");
    }

    public void put(String name, String value) {
        SsmClient ssmClient = SsmClient.builder()
            .region(Region.US_EAST_1)
            .build();

        PutParameterRequest parameterRequest = PutParameterRequest.builder()
            .name(name)
            .value(value)
            .overwrite(true)
            .type("String")
            .build();
    }
}

```

```
        ssmClient.putParameter(parameterRequest);
        System.out.printf("Setting demo parameter %s to '%s'.", name, value);
    }
}
```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella Documentazione di riferimento delle API AWS SDK for Java 2.x .

- [AttachLoadBalancerTargetGroups](#)
- [CreateAutoScalingGroup](#)
- [CreateInstanceProfile](#)
- [CreateLaunchTemplate](#)
- [CreateListener](#)
- [CreateLoadBalancer](#)
- [CreateTargetGroup](#)
- [DeleteAutoScalingGroup](#)
- [DeleteInstanceProfile](#)
- [DeleteLaunchTemplate](#)
- [DeleteLoadBalancer](#)
- [DeleteTargetGroup](#)
- [DescribeAutoScalingGroups](#)
- [DescribeAvailabilityZones](#)
- [DescribeIamInstanceProfileAssociations](#)
- [DescribeInstances](#)
- [DescribeLoadBalancers](#)
- [DescribeSubnets](#)
- [DescribeTargetGroups](#)
- [DescribeTargetHealth](#)
- [DescribeVpcs](#)
- [RebootInstances](#)

- [ReplaceIamInstanceProfileAssociation](#)
- [TerminateInstanceInAutoScalingGroup](#)

- [UpdateAutoScalingGroup](#)

## MediaStore esempi che utilizzano SDK for Java 2.x

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando AWS SDK for Java 2.x with MediaStore.

Le operazioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le operazioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati e negli esempi tra servizi.

Scenari: esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio.

Ogni esempio include un collegamento a GitHub, dove è possibile trovare istruzioni su come configurare ed eseguire il codice nel contesto.

### Argomenti

- [Azioni](#)

### Azioni

#### **CreateContainer**

Il seguente esempio di codice mostra come utilizzare `CreateContainer`.

SDK per Java 2.x

#### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.services.mediastore.MediaStoreClient;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.mediastore.model.CreateContainerRequest;
import software.amazon.awssdk.services.mediastore.model.CreateContainerResponse;
import software.amazon.awssdk.services.mediastore.model.MediaStoreException;
```

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateContainer {
    public static long sleepTime = 10;

    public static void main(String[] args) {
        final String usage = ""

            Usage:    <containerName>

            Where:
                containerName - The name of the container to create.
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String containerName = args[0];
        Region region = Region.US_EAST_1;
        MediaStoreClient mediaStoreClient = MediaStoreClient.builder()
            .region(region)
            .build();

        createMediaContainer(mediaStoreClient, containerName);
        mediaStoreClient.close();
    }

    public static void createMediaContainer(MediaStoreClient mediaStoreClient,
        String containerName) {
        try {
            CreateContainerRequest containerRequest =
            CreateContainerRequest.builder()
                .containerName(containerName)
                .build();
        }
    }
}
```



```
        CreateContainerResponse containerResponse =
mediaStoreClient.createContainer(containerRequest);
        String status = containerResponse.container().status().toString();
        while (!status.equalsIgnoreCase("Active")) {
            status = DescribeContainer.checkContainer(mediaStoreClient,
containerName);
            System.out.println("Status - " + status);
            Thread.sleep(sleepTime * 1000);
        }

        System.out.println("The container ARN value is " +
containerResponse.container().arn());
        System.out.println("Finished ");

    } catch (MediaStoreException | InterruptedException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
}
```

- Per i dettagli sull'API, [CreateContainer](#) consulta AWS SDK for Java 2.x API Reference.

## DeleteContainer

Il seguente esempio di codice mostra come utilizzare `DeleteContainer`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.services.mediastore.MediaStoreClient;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.mediastore.model.CreateContainerRequest;
import software.amazon.awssdk.services.mediastore.model.CreateContainerResponse;
import software.amazon.awssdk.services.mediastore.model.MediaStoreException;
```

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateContainer {
    public static long sleepTime = 10;

    public static void main(String[] args) {
        final String usage = ""

            Usage:    <containerName>

            Where:
                containerName - The name of the container to create.
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String containerName = args[0];
        Region region = Region.US_EAST_1;
        MediaStoreClient mediaStoreClient = MediaStoreClient.builder()
            .region(region)
            .build();

        createMediaContainer(mediaStoreClient, containerName);
        mediaStoreClient.close();
    }

    public static void createMediaContainer(MediaStoreClient mediaStoreClient,
        String containerName) {
        try {
            CreateContainerRequest containerRequest =
            CreateContainerRequest.builder()
                .containerName(containerName)
                .build();
        }
    }
}
```

```

        CreateContainerResponse containerResponse =
mediaStoreClient.createContainer(containerRequest);
        String status = containerResponse.container().status().toString();
        while (!status.equalsIgnoreCase("Active")) {
            status = DescribeContainer.checkContainer(mediaStoreClient,
containerName);
            System.out.println("Status - " + status);
            Thread.sleep(sleepTime * 1000);
        }

        System.out.println("The container ARN value is " +
containerResponse.container().arn());
        System.out.println("Finished ");

    } catch (MediaStoreException | InterruptedException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
}
}

```

- Per i dettagli sull'API, [DeleteContainer](#) consulta AWS SDK for Java 2.x API Reference.

## DeleteObject

Il seguente esempio di codice mostra come utilizzare `DeleteObject`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.mediastore.MediaStoreClient;
import software.amazon.awssdk.services.mediastore.model.DescribeContainerRequest;
import software.amazon.awssdk.services.mediastore.model.DescribeContainerResponse;
import software.amazon.awssdk.services.mediastoredata.MediaStoreDataClient;
import software.amazon.awssdk.services.mediastoredata.model.DeleteObjectRequest;

```

```
import software.amazon.awssdk.services.mediastoredata.model.MediaStoreDataException;
import java.net.URI;
import java.net.URISyntaxException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteObject {
    public static void main(String[] args) throws URISyntaxException {
        final String usage = ""

                Usage:    <completePath> <containerName>

                Where:
                    completePath - The path (including the container) of the item to
delete.
                    containerName - The name of the container.
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String completePath = args[0];
        String containerName = args[1];
        Region region = Region.US_EAST_1;
        URI uri = new URI(getEndpoint(containerName));

        MediaStoreDataClient mediaStoreData = MediaStoreDataClient.builder()
            .endpointOverride(uri)
            .region(region)
            .build();

        deleteMediaObject(mediaStoreData, completePath);
        mediaStoreData.close();
    }
}
```

```
public static void deleteMediaObject(MediaStoreDataClient mediaStoreData, String
completePath) {
    try {
        DeleteObjectRequest deleteObjectRequest = DeleteObjectRequest.builder()
            .path(completePath)
            .build();

        mediaStoreData.deleteObject(deleteObjectRequest);

    } catch (MediaStoreDataException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

private static String getEndpoint(String containerName) {
    Region region = Region.US_EAST_1;
    MediaStoreClient mediaStoreClient = MediaStoreClient.builder()
        .region(region)
        .build();

    DescribeContainerRequest containerRequest =
DescribeContainerRequest.builder()
    .containerName(containerName)
    .build();

    DescribeContainerResponse response =
mediaStoreClient.describeContainer(containerRequest);
    mediaStoreClient.close();
    return response.container().endpoint();
}
}
```

- Per i dettagli sull'API, [DeleteObject](#) consulta AWS SDK for Java 2.x API Reference.

## DescribeContainer

Il seguente esempio di codice mostra come utilizzare `DescribeContainer`.

## SDK per Java 2.x

 Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.mediastore.MediaStoreClient;
import software.amazon.awssdk.services.mediastore.model.DescribeContainerRequest;
import software.amazon.awssdk.services.mediastore.model.DescribeContainerResponse;
import software.amazon.awssdk.services.mediastore.model.MediaStoreException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DescribeContainer {

    public static void main(String[] args) {
        final String usage = ""

            Usage:    <containerName>

            Where:
                containerName - The name of the container to describe.
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String containerName = args[0];
        Region region = Region.US_EAST_1;
        MediaStoreClient mediaStoreClient = MediaStoreClient.builder()
            .region(region)
```

```
        .build();

        System.out.println("Status is " + checkContainer(mediaStoreClient,
containerName));
        mediaStoreClient.close();
    }

    public static String checkContainer(MediaStoreClient mediaStoreClient, String
containerName) {
        try {
            DescribeContainerRequest describeContainerRequest =
DescribeContainerRequest.builder()
                .containerName(containerName)
                .build();

            DescribeContainerResponse containerResponse =
mediaStoreClient.describeContainer(describeContainerRequest);
            System.out.println("The container name is " +
containerResponse.container().name());
            System.out.println("The container ARN is " +
containerResponse.container().arn());
            return containerResponse.container().status().toString();

        } catch (MediaStoreException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
}
```

- Per i dettagli sull'API, [DescribeContainer](#) consulta AWS SDK for Java 2.x API Reference.

## GetObject

Il seguente esempio di codice mostra come utilizzare `GetObject`.

## SDK per Java 2.x

 Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.core.ResponseInputStream;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.mediastore.MediaStoreClient;
import software.amazon.awssdk.services.mediastore.model.DescribeContainerRequest;
import software.amazon.awssdk.services.mediastore.model.DescribeContainerResponse;
import software.amazon.awssdk.services.mediastoredata.MediaStoreDataClient;
import software.amazon.awssdk.services.mediastoredata.model.GetObjectRequest;
import software.amazon.awssdk.services.mediastoredata.model.GetObjectResponse;
import software.amazon.awssdk.services.mediastoredata.model.MediaStoreDataException;
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.OutputStream;
import java.net.URI;
import java.net.URISyntaxException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class GetObject {
    public static void main(String[] args) throws URISyntaxException {
        final String usage = ""

                Usage:    <completePath> <containerName> <savePath>

                Where:
                    completePath - The path of the object in the container (for
example, Videos5/sampleVideo.mp4).
                    containerName - The name of the container.
```



```
        savePath - The path on the local drive where the file is saved,
including the file name (for example, C:/AWS/myvid.mp4).
        """;

    if (args.length != 3) {
        System.out.println(usage);
        System.exit(1);
    }

    String completePath = args[0];
    String containerName = args[1];
    String savePath = args[2];

    Region region = Region.US_EAST_1;
    URI uri = new URI(getEndpoint(containerName));
    MediaStoreDataClient mediaStoreData = MediaStoreDataClient.builder()
        .endpointOverride(uri)
        .region(region)
        .build();

    getMediaObject(mediaStoreData, completePath, savePath);
    mediaStoreData.close();
}

public static void getMediaObject(MediaStoreDataClient mediaStoreData, String
completePath, String savePath) {

    try {
        GetObjectRequest objectRequest = GetObjectRequest.builder()
            .path(completePath)
            .build();

        // Write out the data to a file.
        ResponseInputStream<GetObjectResponse> data =
mediaStoreData.getObject(objectRequest);
        byte[] buffer = new byte[data.available()];
        data.read(buffer);

        File targetFile = new File(savePath);
        OutputStream outputStream = new FileOutputStream(targetFile);
        outputStream.write(buffer);
        System.out.println("The data was written to " + savePath);

    } catch (MediaStoreDataException | IOException e) {
```

```
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

private static String getEndpoint(String containerName) {
    Region region = Region.US_EAST_1;
    MediaStoreClient mediaStoreClient = MediaStoreClient.builder()
        .region(region)
        .build();

    DescribeContainerRequest containerRequest =
DescribeContainerRequest.builder()
        .containerName(containerName)
        .build();

    DescribeContainerResponse response =
mediaStoreClient.describeContainer(containerRequest);
    return response.container().endpoint();
}
}
```

- Per i dettagli sull'API, [GetObject](#) consulta AWS SDK for Java 2.x API Reference.

## ListContainers

Il seguente esempio di codice mostra come utilizzare `ListContainers`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.mediastore.MediaStoreClient;
import software.amazon.awssdk.services.mediastore.model.Container;
```

```
import software.amazon.awssdk.services.mediastore.model.ListContainersResponse;
import software.amazon.awssdk.services.mediastore.model.MediaStoreException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListContainers {

    public static void main(String[] args) {

        Region region = Region.US_EAST_1;
        MediaStoreClient mediaStoreClient = MediaStoreClient.builder()
            .region(region)
            .build();

        listAllContainers(mediaStoreClient);
        mediaStoreClient.close();
    }

    public static void listAllContainers(MediaStoreClient mediaStoreClient) {
        try {
            ListContainersResponse containersResponse =
mediaStoreClient.listContainers();
            List<Container> containers = containersResponse.containers();
            for (Container container : containers) {
                System.out.println("Container name is " + container.name());
            }

        } catch (MediaStoreException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Per i dettagli sull'API, [ListContainers](#) consulta AWS SDK for Java 2.x API Reference.

## PutObject

Il seguente esempio di codice mostra come utilizzare `PutObject`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.mediastore.MediaStoreClient;
import software.amazon.awssdk.services.mediastoredata.MediaStoreDataClient;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.services.mediastoredata.model.PutObjectRequest;
import software.amazon.awssdk.services.mediastoredata.model.MediaStoreDataException;
import software.amazon.awssdk.services.mediastoredata.model.PutObjectResponse;
import software.amazon.awssdk.services.mediastore.model.DescribeContainerRequest;
import software.amazon.awssdk.services.mediastore.model.DescribeContainerResponse;
import java.io.File;
import java.net.URI;
import java.net.URISyntaxException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class PutObject {
    public static void main(String[] args) throws URISyntaxException {
        final String USAGE = ""

                "To run this example, supply the name of a container, a file location\n"
                "to use, and path in the container\s

                "Ex: <containerName> <filePath> <completePath>\n"
                """;
    }
}
```

```
    if (args.length < 3) {
        System.out.println(USAGE);
        System.exit(1);
    }

    String containerName = args[0];
    String filePath = args[1];
    String completePath = args[2];

    Region region = Region.US_EAST_1;
    URI uri = new URI(getEndpoint(containerName));
    MediaStoreDataClient mediaStoreData = MediaStoreDataClient.builder()
        .endpointOverride(uri)
        .region(region)
        .build();

    putMediaObject(mediaStoreData, filePath, completePath);
    mediaStoreData.close();
}

public static void putMediaObject(MediaStoreDataClient mediaStoreData, String
filePath, String completePath) {
    try {
        File myFile = new File(filePath);
        RequestBody requestBody = RequestBody.fromFile(myFile);

        PutObjectRequest objectRequest = PutObjectRequest.builder()
            .path(completePath)
            .contentType("video/mp4")
            .build();

        PutObjectResponse response = mediaStoreData.putObject(objectRequest,
requestBody);
        System.out.println("The saved object is " +
response.storageClass().toString());

    } catch (MediaStoreDataException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static String getEndpoint(String containerName) {
```

```
Region region = Region.US_EAST_1;
MediaStoreClient mediaStoreClient = MediaStoreClient.builder()
    .region(region)
    .build();

DescribeContainerRequest containerRequest =
DescribeContainerRequest.builder()
    .containerName(containerName)
    .build();

DescribeContainerResponse response =
mediaStoreClient.describeContainer(containerRequest);
return response.container().endpoint();
}
}
```

- Per i dettagli sull'API, [PutObject](#) consulta AWS SDK for Java 2.x API Reference.

## OpenSearch Esempi di servizi che utilizzano SDK for Java 2.x

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando AWS SDK for Java 2.x with OpenSearch Service.

Le operazioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le operazioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati e negli esempi tra servizi.

Scenari: esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio.

Ogni esempio include un collegamento a GitHub, dove è possibile trovare istruzioni su come configurare ed eseguire il codice nel contesto.

### Argomenti

- [Azioni](#)

## Azioni

### CreateDomain

Il seguente esempio di codice mostra come utilizzare `CreateDomain`.

SDK per Java 2.x

#### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.opensearch.OpenSearchClient;
import software.amazon.awssdk.services.opensearch.model.ClusterConfig;
import software.amazon.awssdk.services.opensearch.model.EBSOptions;
import software.amazon.awssdk.services.opensearch.model.VolumeType;
import software.amazon.awssdk.services.opensearch.model.NodeToNodeEncryptionOptions;
import software.amazon.awssdk.services.opensearch.model.CreateDomainRequest;
import software.amazon.awssdk.services.opensearch.model.CreateDomainResponse;
import software.amazon.awssdk.services.opensearch.model.OpenSearchException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateDomain {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <domainName>

                Where:
                domainName - The name of the domain to create.
                """;
```

```
        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String domainName = args[0];
        Region region = Region.US_EAST_1;
        OpenSearchClient searchClient = OpenSearchClient.builder()
            .region(region)
            .build();

        createNewDomain(searchClient, domainName);
        System.out.println("Done");
    }

    public static void createNewDomain(OpenSearchClient searchClient, String
domainName) {
        try {
            ClusterConfig clusterConfig = ClusterConfig.builder()
                .dedicatedMasterEnabled(true)
                .dedicatedMasterCount(3)
                .dedicatedMasterType("t2.small.search")
                .instanceType("t2.small.search")
                .instanceCount(5)
                .build();

            EBSOptions ebsOptions = EBSOptions.builder()
                .ebsEnabled(true)
                .volumeSize(10)
                .volumeType(VolumeType.GP2)
                .build();

            NodeToNodeEncryptionOptions encryptionOptions =
NodeToNodeEncryptionOptions.builder()
                .enabled(true)
                .build();

            CreateDomainRequest domainRequest = CreateDomainRequest.builder()
                .domainName(domainName)
                .engineVersion("OpenSearch_1.0")
                .clusterConfig(clusterConfig)
                .ebsOptions(ebsOptions)
                .nodeToNodeEncryptionOptions(encryptionOptions)
                .build();
```



```
        System.out.println("Sending domain creation request...");
        CreateDomainResponse createResponse =
searchClient.createDomain(domainRequest);
        System.out.println("Domain status is " +
createResponse.domainStatus().toString());
        System.out.println("Domain Id is " +
createResponse.domainStatus().domainId());

        } catch (OpenSearchException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Per i dettagli sull'API, [CreateDomain](#) consulta AWS SDK for Java 2.x API Reference.

## DeleteDomain

Il seguente esempio di codice mostra come utilizzare `DeleteDomain`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.opensearch.OpenSearchClient;
import software.amazon.awssdk.services.opensearch.model.OpenSearchException;
import software.amazon.awssdk.services.opensearch.model.DeleteDomainRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 */
```

```
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class DeleteDomain {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <domainName>

            Where:
                domainName - The name of the domain to delete.
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String domainName = args[0];
        Region region = Region.US_EAST_1;
        OpenSearchClient searchClient = OpenSearchClient.builder()
            .region(region)
            .build();

        deleteSpecificDomain(searchClient, domainName);
        System.out.println("Done");
    }

    public static void deleteSpecificDomain(OpenSearchClient searchClient, String
domainName) {
        try {
            DeleteDomainRequest domainRequest = DeleteDomainRequest.builder()
                .domainName(domainName)
                .build();

            searchClient.deleteDomain(domainRequest);
            System.out.println(domainName + " was successfully deleted.");

        } catch (OpenSearchException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Per i dettagli sull'API, [DeleteDomain](#) consulta AWS SDK for Java 2.x API Reference.

## ListDomainNames

Il seguente esempio di codice mostra come utilizzare `ListDomainNames`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.opensearch.OpenSearchClient;
import software.amazon.awssdk.services.opensearch.model.DomainInfo;
import software.amazon.awssdk.services.opensearch.model.ListDomainNamesRequest;
import software.amazon.awssdk.services.opensearch.model.ListDomainNamesResponse;
import software.amazon.awssdk.services.opensearch.model.OpenSearchException;

import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListDomainNames {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        OpenSearchClient searchClient = OpenSearchClient.builder()
            .region(region)
            .credentialsProvider(ProfileCredentialsProvider.create())
            .build();
        listAllDomains(searchClient);
    }
}
```

```
        System.out.println("Done");
    }

    public static void listAllDomains(OpenSearchClient searchClient) {
        try {
            ListDomainNamesRequest namesRequest = ListDomainNamesRequest.builder()
                .engineType("OpenSearch")
                .build();

            ListDomainNamesResponse response =
searchClient.listDomainNames(namesRequest);
            List<DomainInfo> domainInfoList = response.domainNames();
            for (DomainInfo domain : domainInfoList)
                System.out.println("Domain name is " + domain.domainName());

        } catch (OpenSearchException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Per i dettagli sull'API, [ListDomainNames](#) consulta AWS SDK for Java 2.x API Reference.

## UpdateDomainConfig

Il seguente esempio di codice mostra come utilizzare `UpdateDomainConfig`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.opensearch.OpenSearchClient;
import software.amazon.awssdk.services.opensearch.model.ClusterConfig;
import software.amazon.awssdk.services.opensearch.model.OpenSearchException;
import software.amazon.awssdk.services.opensearch.model.UpdateDomainConfigRequest;
```

```
import software.amazon.awssdk.services.opensearch.model.UpdateDomainConfigResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class UpdateDomain {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <domainName>

                Where:
                domainName - The name of the domain to update.

                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String domainName = args[0];
        Region region = Region.US_EAST_1;
        OpenSearchClient searchClient = OpenSearchClient.builder()
                .region(region)
                .build();

        updateSpecificDomain(searchClient, domainName);
        System.out.println("Done");
    }

    public static void updateSpecificDomain(OpenSearchClient searchClient, String
domainName) {
        try {
            ClusterConfig clusterConfig = ClusterConfig.builder()
                    .instanceCount(3)
                    .build();

```

```
UpdateDomainConfigRequest updateDomainConfigRequest =
UpdateDomainConfigRequest.builder()
    .domainName(domainName)
    .clusterConfig(clusterConfig)
    .build();

System.out.println("Sending domain update request...");
UpdateDomainConfigResponse updateResponse =
searchClient.updateDomainConfig(updateDomainConfigRequest);
System.out.println("Domain update response from Amazon OpenSearch
Service:");
System.out.println(updateResponse.toString());

    } catch (OpenSearchException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Per i dettagli sull'API, [UpdateDomainConfig](#) consulta AWS SDK for Java 2.x API Reference.

## EventBridge esempi che utilizzano SDK for Java 2.x

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando AWS SDK for Java 2.x with EventBridge.

Le operazioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le operazioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati e negli esempi tra servizi.

Scenari: esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio.

Ogni esempio include un collegamento a GitHub, dove è possibile trovare istruzioni su come configurare ed eseguire il codice nel contesto.

Nozioni di base

## Salve EventBridge

I seguenti esempi di codice mostrano come iniziare a utilizzare EventBridge.

### SDK per Java 2.x

#### Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class HelloEventBridge {
    public static void main(String[] args) {
        Region region = Region.US_WEST_2;
        EventBridgeClient eventBrClient = EventBridgeClient.builder()
            .region(region)
            .build();

        listBuses(eventBrClient);
        eventBrClient.close();
    }

    public static void listBuses(EventBridgeClient eventBrClient) {
        try {
            ListEventBusesRequest busesRequest = ListEventBusesRequest.builder()
                .limit(10)
                .build();

            ListEventBusesResponse response =
eventBrClient.listEventBuses(busesRequest);
            List<EventBus> buses = response.eventBuses();
            for (EventBus bus : buses) {
                System.out.println("The name of the event bus is: " + bus.name());
            }
        }
    }
}
```

```
        System.out.println("The ARN of the event bus is: " + bus.arn());
    }

    } catch (EventBridgeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Per i dettagli sull'API, [ListEventBuses](#) consulta AWS SDK for Java 2.x API Reference.

## Argomenti

- [Azioni](#)
- [Scenari](#)

## Azioni

### DeleteRule

Il seguente esempio di codice mostra come utilizzare `DeleteRule`.

SDK per Java 2.x

#### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public static void deleteRuleByName(EventBridgeClient eventBrClient, String
ruleName) {
    DeleteRuleRequest ruleRequest = DeleteRuleRequest.builder()
        .name(ruleName)
        .build();

    eventBrClient.deleteRule(ruleRequest);
    System.out.println("Successfully deleted the rule");
}
```



```
}
```

- Per i dettagli sull'API, [DeleteRule](#) consulta AWS SDK for Java 2.x API Reference.

## DescribeRule

Il seguente esempio di codice mostra come utilizzare `DescribeRule`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public static void checkRule(EventBridgeClient eventBrClient, String
eventRuleName) {
    try {
        DescribeRuleRequest ruleRequest = DescribeRuleRequest.builder()
            .name(eventRuleName)
            .build();

        DescribeRuleResponse response = eventBrClient.describeRule(ruleRequest);
        System.out.println("The state of the rule is " +
response.stateAsString());

    } catch (EventBridgeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Per i dettagli sull'API, [DescribeRule](#) consulta AWS SDK for Java 2.x API Reference.

## DisableRule

Il seguente esempio di codice mostra come utilizzare `DisableRule`.

## SDK per Java 2.x

 Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Disabilita una regola utilizzando il nome della stessa.

```
public static void changeRuleState(EventBridgeClient eventBrClient, String
eventRuleName, Boolean isEnabled) {
    try {
        if (!isEnabled) {
            System.out.println("Disabling the rule: " + eventRuleName);
            DisableRuleRequest ruleRequest = DisableRuleRequest.builder()
                .name(eventRuleName)
                .build();

            eventBrClient.disableRule(ruleRequest);
        } else {
            System.out.println("Enabling the rule: " + eventRuleName);
            EnableRuleRequest ruleRequest = EnableRuleRequest.builder()
                .name(eventRuleName)
                .build();
            eventBrClient.enableRule(ruleRequest);
        }
    } catch (EventBridgeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Per i dettagli sull'API, [DisableRule](#) consulta AWS SDK for Java 2.x API Reference.

## EnableRule

Il seguente esempio di codice mostra come utilizzare `EnableRule`.

## SDK per Java 2.x

 Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Abilita una regola utilizzando il nome della stessa.

```
public static void changeRuleState(EventBridgeClient eventBrClient, String
eventRuleName, Boolean isEnabled) {
    try {
        if (!isEnabled) {
            System.out.println("Disabling the rule: " + eventRuleName);
            DisableRuleRequest ruleRequest = DisableRuleRequest.builder()
                .name(eventRuleName)
                .build();


            eventBrClient.disableRule(ruleRequest);
        } else {
            System.out.println("Enabling the rule: " + eventRuleName);
            EnableRuleRequest ruleRequest = EnableRuleRequest.builder()
                .name(eventRuleName)
                .build();
            eventBrClient.enableRule(ruleRequest);
        }
    } catch (EventBridgeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Per i dettagli sull'API, [EnableRule](#) consulta AWS SDK for Java 2.x API Reference.

## ListRuleNamesByTarget

Il seguente esempio di codice mostra come utilizzare `ListRuleNamesByTarget`.

## SDK per Java 2.x

 Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Elenca tutti i nomi delle regole utilizzando la destinazione.

```
public static void listTargetRules(EventBridgeClient eventBrClient, String
topicArn) {
    ListRuleNamesByTargetRequest ruleNamesByTargetRequest =
ListRuleNamesByTargetRequest.builder()
        .targetArn(topicArn)
        .build();


    ListRuleNamesByTargetResponse response =
eventBrClient.listRuleNamesByTarget(ruleNamesByTargetRequest);
    List<String> rules = response.ruleNames();
    for (String rule : rules) {
        System.out.println("The rule name is " + rule);
    }
}
```

- Per i dettagli sull'API, [ListRuleNamesByTarget](#) consulta AWS SDK for Java 2.x API Reference.

## ListRules

Il seguente esempio di codice mostra come utilizzare `ListRules`.

## SDK per Java 2.x

 Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Abilita una regola utilizzando il nome della stessa.

```
public static void listRules(EventBridgeClient eventBrClient) {
    try {
        ListRulesRequest rulesRequest = ListRulesRequest.builder()
            .eventBusName("default")
            .limit(10)
            .build();

        ListRulesResponse response = eventBrClient.listRules(rulesRequest);
        List<Rule> rules = response.rules();
        for (Rule rule : rules) {
            System.out.println("The rule name is : " + rule.name());
            System.out.println("The rule description is : " +
rule.description());
            System.out.println("The rule state is : " + rule.stateAsString());
        }

    } catch (EventBridgeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Per i dettagli sull'API, [ListRules](#) consulta AWS SDK for Java 2.x API Reference.

## ListTargetsByRule

Il seguente esempio di codice mostra come utilizzare `ListTargetsByRule`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Elenca tutte le destinazioni di una regola utilizzando il nome della stessa.

```
public static void listTargets(EventBridgeClient eventBrClient, String ruleName)
{
```

```

ListTargetsByRuleRequest ruleRequest = ListTargetsByRuleRequest.builder()
    .rule(ruleName)
    .build();

ListTargetsByRuleResponse res =
eventBrClient.listTargetsByRule(ruleRequest);
List<Target> targetsList = res.targets();
for (Target target: targetsList) {
    System.out.println("Target ARN: "+target.arn());
}
}

```

- Per i dettagli sull'API, [ListTargetsByRule](#) consulta AWS SDK for Java 2.x API Reference.

## PutEvents

Il seguente esempio di codice mostra come utilizzare PutEvents.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```

public static void triggerCustomRule(EventBridgeClient eventBrClient, String
email) {
    String json = "{" +
        "\"UserEmail\": \"" + email + "\", " +
        "\"Message\": \"This event was generated by example code.\", " +
        "\"UtcTime\": \"Now.\" " +
        "}";

    PutEventsRequestEntry entry = PutEventsRequestEntry.builder()
        .source("ExampleSource")
        .detail(json)
        .detailType("ExampleType")
        .build();
}

```

```
PutEventsRequest eventsRequest = PutEventsRequest.builder()
    .entries(entry)
    .build();

eventBrClient.putEvents(eventsRequest);
}
```

- Per i dettagli sull'API, [PutEvents](#) consulta AWS SDK for Java 2.x API Reference.

## PutRule

Il seguente esempio di codice mostra come utilizzare `PutRule`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Crea una regola pianificata.

```
public static void createEBRule(EventBridgeClient eventBrClient, String
ruleName, String cronExpression) {
    try {
        PutRuleRequest ruleRequest = PutRuleRequest.builder()
            .name(ruleName)
            .eventBusName("default")
            .scheduleExpression(cronExpression)
            .state("ENABLED")
            .description("A test rule that runs on a schedule created by the
Java API")
            .build();

        PutRuleResponse ruleResponse = eventBrClient.putRule(ruleRequest);
        System.out.println("The ARN of the new rule is " +
ruleResponse.ruleArn());

    } catch (EventBridgeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }
}
```

```

        System.exit(1);
    }
}

```

Crea una regola che si attiva quando un oggetto viene aggiunto a un bucket di Amazon Simple Storage Service.

```

// Create a new event rule that triggers when an Amazon S3 object is created in
// a bucket.
public static void addEventRule(EventBridgeClient eventBrClient, String roleArn,
String bucketName,
    String eventRuleName) {
    String pattern = "{\n" +
        "  \"source\": [\"aws.s3\"],\n" +
        "  \"detail-type\": [\"Object Created\"],\n" +
        "  \"detail\": {\n" +
        "    \"bucket\": {\n" +
        "      \"name\": [\"\" + bucketName + "\"]\n" +
        "    }\n" +
        "  }\n" +
        "}";

    try {
        PutRuleRequest ruleRequest = PutRuleRequest.builder()
            .description("Created by using the AWS SDK for Java v2")
            .name(eventRuleName)
            .eventPattern(pattern)
            .roleArn(roleArn)
            .build();

        PutRuleResponse ruleResponse = eventBrClient.putRule(ruleRequest);
        System.out.println("The ARN of the new rule is " +
ruleResponse.ruleArn());

    } catch (EventBridgeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

```

- Per i dettagli sull'API, [PutRule](#) consulta AWS SDK for Java 2.x API Reference.



## PutTargets

Il seguente esempio di codice mostra come utilizzare PutTargets.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Aggiungi un argomento Amazon SNS come destinazione per una regola.

```
// Add a rule which triggers an SNS target when a file is uploaded to an S3
// bucket.
public static void addSnsEventRule(EventBridgeClient eventBrClient, String
ruleName, String topicArn,
    String topicName, String eventRuleName, String bucketName) {
    String targetID = java.util.UUID.randomUUID().toString();
    Target myTarget = Target.builder()
        .id(targetID)
        .arn(topicArn)
        .build();

    List<Target> targets = new ArrayList<>();
    targets.add(myTarget);
    PutTargetsRequest request = PutTargetsRequest.builder()
        .eventBusName(null)
        .targets(targets)
        .rule(ruleName)
        .build();

    eventBrClient.putTargets(request);
    System.out.println("Added event rule " + eventRuleName + " with Amazon SNS
target " + topicName + " for bucket "
        + bucketName + ".");
}
```

Aggiungi un trasformatore di input a una destinazione per una regola.

```
public static void updateCustomRuleTargetWithTransform(EventBridgeClient
eventBrClient, String topicArn,
    String ruleName) {
    String targetId = java.util.UUID.randomUUID().toString();
    InputTransformer inputTransformer = InputTransformer.builder()
        .inputTemplate("\Notification: sample event was received.\")
        .build();

    Target target = Target.builder()
        .id(targetId)
        .arn(topicArn)
        .inputTransformer(inputTransformer)
        .build();

    try {
        PutTargetsRequest targetsRequest = PutTargetsRequest.builder()
            .rule(ruleName)
            .targets(target)
            .eventBusName(null)
            .build();

        eventBrClient.putTargets(targetsRequest);
    } catch (EventBridgeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Per i dettagli sull'API, [PutTargets](#) consulta AWS SDK for Java 2.x API Reference.

## RemoveTargets

Il seguente esempio di codice mostra come utilizzare `RemoveTargets`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Rimuovi tutte le destinazioni di una regola utilizzando il nome della stessa.

```
public static void deleteTargetsFromRule(EventBridgeClient eventBrClient, String
eventRuleName) {
    // First, get all targets that will be deleted.
    ListTargetsByRuleRequest request = ListTargetsByRuleRequest.builder()
        .rule(eventRuleName)
        .build();

    ListTargetsByRuleResponse response =
eventBrClient.listTargetsByRule(request);
    List<Target> allTargets = response.targets();

    // Get all targets and delete them.
    for (Target myTarget : allTargets) {
        RemoveTargetsRequest removeTargetsRequest =
RemoveTargetsRequest.builder()
            .rule(eventRuleName)
            .ids(myTarget.id())
            .build();

        eventBrClient.removeTargets(removeTargetsRequest);
        System.out.println("Successfully removed the target");
    }
}
```

- Per i dettagli sull'API, [RemoveTargets](#) consulta AWS SDK for Java 2.x API Reference.

## Scenari

Nozioni di base su regole e destinazioni

L'esempio di codice seguente mostra come:

- Creare una regola e aggiungervi una destinazione.
- Abilitare e disabilitare regole.
- Elencare e aggiornare regole e destinazioni.
- Inviare eventi e quindi eliminare le risorse.

## SDK per Java 2.x

 Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * This Java code example performs the following tasks:
 *
 * This Java V2 example performs the following tasks with Amazon EventBridge:
 *
 * 1. Creates an AWS Identity and Access Management (IAM) role to use with
 * Amazon EventBridge.
 * 2. Amazon Simple Storage Service (Amazon S3) bucket with EventBridge events
 * enabled.
 * 3. Creates a rule that triggers when an object is uploaded to Amazon S3.
 * 4. Lists rules on the event bus.
 * 5. Creates a new Amazon Simple Notification Service (Amazon SNS) topic and
 * lets the user subscribe to it.
 * 6. Adds a target to the rule that sends an email to the specified topic.
 * 7. Creates an EventBridge event that sends an email when an Amazon S3 object
 * is created.
 * 8. Lists Targets.
 * 9. Lists the rules for the same target.
 * 10. Triggers the rule by uploading a file to the Amazon S3 bucket.
 * 11. Disables a specific rule.
 * 12. Checks and print the state of the rule.
 * 13. Adds a transform to the rule to change the text of the email.
 * 14. Enables a specific rule.
 * 15. Triggers the updated rule by uploading a file to the Amazon S3 bucket.
 * 16. Updates the rule to be a custom rule pattern.
 * 17. Sending an event to trigger the rule.
 * 18. Cleans up resources.
```

```

*
*/
public class EventbridgeMVP {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    public static void main(String[] args) throws InterruptedException, IOException
    {
        final String usage = ""

            Usage:
                <roleName> <bucketName> <topicName> <eventRuleName>

            Where:
                roleName - The name of the role to create.
                bucketName - The Amazon Simple Storage Service (Amazon S3)
bucket name to create.
                topicName - The name of the Amazon Simple Notification Service
(Amazon SNS) topic to create.
                eventRuleName - The Amazon EventBridge rule name to create.
            """;

        if (args.length != 5) {
            System.out.println(usage);
            System.exit(1);
        }

        String polJSON = "{" +
            "\"Version\": \"2012-10-17\"," +
            "\"Statement\": [{" +
            "\"Effect\": \"Allow\"," +
            "\"Principal\": {" +
            "\"Service\": \"events.amazonaws.com\"" +
            "}," +
            "\"Action\": \"sts:AssumeRole\"" +
            "}] " +
            "}";

        Scanner sc = new Scanner(System.in);
        String roleName = args[0];
        String bucketName = args[1];
        String topicName = args[2];
        String eventRuleName = args[3];

        Region region = Region.US_EAST_1;

```

```
EventBridgeClient eventBrClient = EventBridgeClient.builder()
    .region(region)
    .build();

S3Client s3Client = S3Client.builder()
    .region(region)
    .build();

Region regionGl = Region.AWS_GLOBAL;
IamClient iam = IamClient.builder()
    .region(regionGl)
    .build();

SnsClient snsClient = SnsClient.builder()
    .region(region)
    .build();

System.out.println(DASHES);
System.out.println("Welcome to the Amazon EventBridge example scenario.");
System.out.println(DASHES);

System.out.println(DASHES);
System.out
    .println("1. Create an AWS Identity and Access Management (IAM) role
to use with Amazon EventBridge.");
String roleArn = createIAMRole(iam, roleName, polJSON);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("2. Create an S3 bucket with EventBridge events
enabled.");
if (checkBucket(s3Client, bucketName)) {
    System.out.println("Bucket " + bucketName + " already exists. Ending
this scenario.");
    System.exit(1);
}

createBucket(s3Client, bucketName);
Thread.sleep(3000);
setBucketNotification(s3Client, bucketName);
System.out.println(DASHES);

System.out.println(DASHES);
```

```
        System.out.println("3. Create a rule that triggers when an object is
uploaded to Amazon S3.");
        Thread.sleep(10000);
        addEventRule(eventBrClient, roleArn, bucketName, eventRuleName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. List rules on the event bus.");
        listRules(eventBrClient);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. Create a new SNS topic for testing and let the user
subscribe to the topic.");
        String topicArn = createSnsTopic(snsClient, topicName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6. Add a target to the rule that sends an email to the
specified topic.");
        System.out.println("Enter your email to subscribe to the Amazon SNS
topic:");
        String email = sc.nextLine();
        subEmail(snsClient, topicArn, email);
        System.out.println(
            "Use the link in the email you received to confirm your
subscription. Then, press Enter to continue.");
        sc.nextLine();
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Create an EventBridge event that sends an email when
an Amazon S3 object is created.");
        addSnsEventRule(eventBrClient, eventRuleName, topicArn, topicName,
eventRuleName, bucketName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println(" 8. List Targets.");
        listTargets(eventBrClient, eventRuleName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println(" 9. List the rules for the same target.");
```

```
listTargetRules(eventBrClient, topicArn);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println(" 10. Trigger the rule by uploading a file to the S3
bucket.");
System.out.println("Press Enter to continue.");
sc.nextLine();
uploadTextFiletoS3(s3Client, bucketName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("11. Disable a specific rule.");
changeRuleState(eventBrClient, eventRuleName, false);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("12. Check and print the state of the rule.");
checkRule(eventBrClient, eventRuleName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("13. Add a transform to the rule to change the text of
the email.");
updateSnsEventRule(eventBrClient, topicArn, eventRuleName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("14. Enable a specific rule.");
changeRuleState(eventBrClient, eventRuleName, true);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println(" 15. Trigger the updated rule by uploading a file to the
S3 bucket.");
System.out.println("Press Enter to continue.");
sc.nextLine();
uploadTextFiletoS3(s3Client, bucketName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println(" 16. Update the rule to be a custom rule pattern.");
updateToCustomRule(eventBrClient, eventRuleName);
```



```

        System.out.println("Updated event rule " + eventRuleName + " to use a custom
pattern.");
        updateCustomRuleTargetWithTransform(eventBrClient, topicArn, eventRuleName);
        System.out.println("Updated event target " + topicArn + ".");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("17. Sending an event to trigger the rule. This will
trigger a subscription email.");
        triggerCustomRule(eventBrClient, email);
        System.out.println("Events have been sent. Press Enter to continue.");
        sc.nextLine();
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("18. Clean up resources.");
        System.out.println("Do you want to clean up resources (y/n)");
        String ans = sc.nextLine();
        if (ans.compareTo("y") == 0) {
            cleanupResources(eventBrClient, snsClient, s3Client, iam, topicArn,
eventRuleName, bucketName, roleName);
        } else {
            System.out.println("The resources will not be cleaned up. ");
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("The Amazon EventBridge example scenario has successfully
completed.");
        System.out.println(DASHES);
    }

    public static void cleanupResources(EventBridgeClient eventBrClient, SnsClient
snsClient, S3Client s3Client,
        IamClient iam, String topicArn, String eventRuleName, String bucketName,
String roleName) {
        System.out.println("Removing all targets from the event rule.");
        deleteTargetsFromRule(eventBrClient, eventRuleName);
        deleteRuleByName(eventBrClient, eventRuleName);
        deleteSNSTopic(snsClient, topicArn);
        deleteS3Bucket(s3Client, bucketName);
        deleteRole(iam, roleName);
    }

```

```
public static void deleteRole(IamClient iam, String roleName) {
    String policyArn = "arn:aws:iam::aws:policy/AmazonEventBridgeFullAccess";
    DetachRolePolicyRequest policyRequest = DetachRolePolicyRequest.builder()
        .policyArn(policyArn)
        .roleName(roleName)
        .build();

    iam.detachRolePolicy(policyRequest);
    System.out.println("Successfully detached policy " + policyArn + " from role
" + roleName);

    // Delete the role.
    DeleteRoleRequest roleRequest = DeleteRoleRequest.builder()
        .roleName(roleName)
        .build();

    iam.deleteRole(roleRequest);
    System.out.println("*** Successfully deleted " + roleName);
}

public static void deleteS3Bucket(S3Client s3Client, String bucketName) {
    // Remove all the objects from the S3 bucket.
    ListObjectsRequest listObjects = ListObjectsRequest.builder()
        .bucket(bucketName)
        .build();

    ListObjectsResponse res = s3Client.listObjects(listObjects);
    List<S3Object> objects = res.contents();
    ArrayList<ObjectIdentifier> toDelete = new ArrayList<>();

    for (S3Object myValue : objects) {
        toDelete.add(ObjectIdentifier.builder()
            .key(myValue.key())
            .build());
    }

    DeleteObjectsRequest dor = DeleteObjectsRequest.builder()
        .bucket(bucketName)
        .delete(Delete.builder()
            .objects(toDelete).build())
        .build();

    s3Client.deleteObjects(dor);
}
```

```
// Delete the S3 bucket.
DeleteBucketRequest deleteBucketRequest = DeleteBucketRequest.builder()
    .bucket(bucketName)
    .build();

s3Client.deleteBucket(deleteBucketRequest);
System.out.println("You have deleted the bucket and the objects");
}

// Delete the SNS topic.
public static void deleteSNSTopic(SnsClient snsClient, String topicArn) {
    try {
        DeleteTopicRequest request = DeleteTopicRequest.builder()
            .topicArn(topicArn)
            .build();

        DeleteTopicResponse result = snsClient.deleteTopic(request);
        System.out.println("\n\nStatus was " +
result.sdkHttpResponse().statusCode());

    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void deleteRuleByName(EventBridgeClient eventBrClient, String
ruleName) {
    DeleteRuleRequest ruleRequest = DeleteRuleRequest.builder()
        .name(ruleName)
        .build();

    eventBrClient.deleteRule(ruleRequest);
    System.out.println("Successfully deleted the rule");
}

public static void deleteTargetsFromRule(EventBridgeClient eventBrClient, String
eventRuleName) {
    // First, get all targets that will be deleted.
    ListTargetsByRuleRequest request = ListTargetsByRuleRequest.builder()
        .rule(eventRuleName)
        .build();
```

```

        ListTargetsByRuleResponse response =
eventBrClient.listTargetsByRule(request);
        List<Target> allTargets = response.targets();

        // Get all targets and delete them.
        for (Target myTarget : allTargets) {
            RemoveTargetsRequest removeTargetsRequest =
RemoveTargetsRequest.builder()
                .rule(eventRuleName)
                .ids(myTarget.id())
                .build();

            eventBrClient.removeTargets(removeTargetsRequest);
            System.out.println("Successfully removed the target");
        }
    }

    public static void triggerCustomRule(EventBridgeClient eventBrClient, String
email) {
        String json = "{" +
            "\"UserEmail\": \"" + email + "\", " +
            "\"Message\": \"This event was generated by example code.\", " +
            "\"UtcTime\": \"Now.\" " +
            "}";

        PutEventsRequestEntry entry = PutEventsRequestEntry.builder()
            .source("ExampleSource")
            .detail(json)
            .detailType("ExampleType")
            .build();

        PutEventsRequest eventsRequest = PutEventsRequest.builder()
            .entries(entry)
            .build();

        eventBrClient.putEvents(eventsRequest);
    }

    public static void updateCustomRuleTargetWithTransform(EventBridgeClient
eventBrClient, String topicArn,
        String ruleName) {
        String targetId = java.util.UUID.randomUUID().toString();
        InputTransformer inputTransformer = InputTransformer.builder()
            .inputTemplate("\"Notification: sample event was received.\"")

```

```
        .build();

    Target target = Target.builder()
        .id(targetId)
        .arn(topicArn)
        .inputTransformer(inputTransformer)
        .build();

    try {
        PutTargetsRequest targetsRequest = PutTargetsRequest.builder()
            .rule(ruleName)
            .targets(target)
            .eventBusName(null)
            .build();

        eventBrClient.putTargets(targetsRequest);
    } catch (EventBridgeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void updateToCustomRule(EventBridgeClient eventBrClient, String
ruleName) {
    String customEventsPattern = "{" +
        "\"source\": [\"ExampleSource\"]," +
        "\"detail-type\": [\"ExampleType\"]" +
        "}";

    PutRuleRequest request = PutRuleRequest.builder()
        .name(ruleName)
        .description("Custom test rule")
        .eventPattern(customEventsPattern)
        .build();

    eventBrClient.putRule(request);
}

// Update an Amazon S3 object created rule with a transform on the target.
public static void updateSnsEventRule(EventBridgeClient eventBrClient, String
topicArn, String ruleName) {
    String targetId = java.util.UUID.randomUUID().toString();
    Map<String, String> myMap = new HashMap<>();
    myMap.put("bucket", "$.detail.bucket.name");
```

```
myMap.put("time", "$.time");

InputTransformer inputTransformer = InputTransformer.builder()
    .inputTemplate("\Notification: an object was uploaded to bucket
<bucket> at <time>.\")
    .inputPathsMap(myMap)
    .build();

Target target = Target.builder()
    .id(targetId)
    .arn(topicArn)
    .inputTransformer(inputTransformer)
    .build();

try {
    PutTargetsRequest targetsRequest = PutTargetsRequest.builder()
        .rule(ruleName)
        .targets(target)
        .eventBusName(null)
        .build();

    eventBrClient.putTargets(targetsRequest);

} catch (EventBridgeException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}

}

public static void checkRule(EventBridgeClient eventBrClient, String
eventRuleName) {
    try {
        DescribeRuleRequest ruleRequest = DescribeRuleRequest.builder()
            .name(eventRuleName)
            .build();

        DescribeRuleResponse response = eventBrClient.describeRule(ruleRequest);
        System.out.println("The state of the rule is " +
response.stateAsString());

    } catch (EventBridgeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

```
    }

    public static void changeRuleState(EventBridgeClient eventBrClient, String
eventRuleName, Boolean isEnabled) {
        try {
            if (!isEnabled) {
                System.out.println("Disabling the rule: " + eventRuleName);
                DisableRuleRequest ruleRequest = DisableRuleRequest.builder()
                    .name(eventRuleName)
                    .build();

                eventBrClient.disableRule(ruleRequest);
            } else {
                System.out.println("Enabling the rule: " + eventRuleName);
                EnableRuleRequest ruleRequest = EnableRuleRequest.builder()
                    .name(eventRuleName)
                    .build();
                eventBrClient.enableRule(ruleRequest);
            }
        } catch (EventBridgeException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    // Create and upload a file to an S3 bucket to trigger an event.
    public static void uploadTextFiletoS3(S3Client s3Client, String bucketName)
throws IOException {
        // Create a unique file name.
        String fileSuffix = new SimpleDateFormat("yyyyMMddHHmmss").format(new
Date());
        String fileName = "TextFile" + fileSuffix + ".txt";

        File myFile = new File(fileName);
        FileWriter fw = new FileWriter(myFile.getAbsolutePath());
        BufferedWriter bw = new BufferedWriter(fw);
        bw.write("This is a sample file for testing uploads.");
        bw.close();

        try {
            PutObjectRequest putObj = PutObjectRequest.builder()
                .bucket(bucketName)
                .key(fileName)
```

```
        .build());

        s3Client.putObject(putOb, RequestBody.fromFile(myFile));

    } catch (S3Exception e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void listTargetRules(EventBridgeClient eventBrClient, String
topicArn) {
    ListRuleNamesByTargetRequest ruleNamesByTargetRequest =
ListRuleNamesByTargetRequest.builder()
        .targetArn(topicArn)
        .build();

    ListRuleNamesByTargetResponse response =
eventBrClient.listRuleNamesByTarget(ruleNamesByTargetRequest);
    List<String> rules = response.ruleNames();
    for (String rule : rules) {
        System.out.println("The rule name is " + rule);
    }
}

public static void listTargets(EventBridgeClient eventBrClient, String ruleName)
{
    ListTargetsByRuleRequest ruleRequest = ListTargetsByRuleRequest.builder()
        .rule(ruleName)
        .build();

    ListTargetsByRuleResponse res =
eventBrClient.listTargetsByRule(ruleRequest);
    List<Target> targetsList = res.targets();
    for (Target target: targetsList) {
        System.out.println("Target ARN: "+target.arn());
    }
}

// Add a rule which triggers an SNS target when a file is uploaded to an S3
// bucket.
public static void addSnsEventRule(EventBridgeClient eventBrClient, String
ruleName, String topicArn,
    String topicName, String eventRuleName, String bucketName) {
```



```
String targetID = java.util.UUID.randomUUID().toString();
Target myTarget = Target.builder()
    .id(targetID)
    .arn(topicArn)
    .build();

List<Target> targets = new ArrayList<>();
targets.add(myTarget);
PutTargetsRequest request = PutTargetsRequest.builder()
    .eventBusName(null)
    .targets(targets)
    .rule(ruleName)
    .build();

eventBrClient.putTargets(request);
System.out.println("Added event rule " + eventRuleName + " with Amazon SNS
target " + topicName + " for bucket "
    + bucketName + ".");
}

public static void subEmail(SnsClient snsClient, String topicArn, String email)
{
    try {
        SubscribeRequest request = SubscribeRequest.builder()
            .protocol("email")
            .endpoint(email)
            .returnSubscriptionArn(true)
            .topicArn(topicArn)
            .build();

        SubscribeResponse result = snsClient.subscribe(request);
        System.out.println("Subscription ARN: " + result.subscriptionArn() + "\n
\n Status is "
            + result.sdkHttpResponse().statusCode());

    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void listRules(EventBridgeClient eventBrClient) {
    try {
        ListRulesRequest rulesRequest = ListRulesRequest.builder()
```

```

        .eventBusName("default")
        .limit(10)
        .build();

ListRulesResponse response = eventBrClient.listRules(rulesRequest);
List<Rule> rules = response.rules();
for (Rule rule : rules) {
    System.out.println("The rule name is : " + rule.name());
    System.out.println("The rule description is : " +
rule.description());
    System.out.println("The rule state is : " + rule.stateAsString());
}

} catch (EventBridgeException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}

public static String createSnsTopic(SnsClient snsClient, String topicName) {
    String topicPolicy = "{" +
        "\"Version\": \"2012-10-17\"," +
        "\"Statement\": [{" +
        "\"Sid\": \"EventBridgePublishTopic\"," +
        "\"Effect\": \"Allow\"," +
        "\"Principal\": {" +
        "\"Service\": \"events.amazonaws.com\"" +
        "}," +
        "\"Resource\": \"*\"," +
        "\"Action\": \"sns:Publish\"" +
        "}]}" +
        "}";

    Map<String, String> topicAttributes = new HashMap<>();
    topicAttributes.put("Policy", topicPolicy);
    CreateTopicRequest topicRequest = CreateTopicRequest.builder()
        .name(topicName)
        .attributes(topicAttributes)
        .build();

    CreateTopicResponse response = snsClient.createTopic(topicRequest);
    System.out.println("Added topic " + topicName + " for email
subscriptions.");
    return response.topicArn();
}

```

```

}

// Create a new event rule that triggers when an Amazon S3 object is created in
// a bucket.
public static void addEventRule(EventBridgeClient eventBrClient, String roleArn,
String bucketName,
    String eventRuleName) {
    String pattern = "{\n" +
        "  \"source\": [\"aws.s3\"],\n" +
        "  \"detail-type\": [\"Object Created\"],\n" +
        "  \"detail\": {\n" +
        "    \"bucket\": {\n" +
        "      \"name\": [\"" + bucketName + "\"]\n" +
        "    }\n" +
        "  }\n" +
        "}";

    try {
        PutRuleRequest ruleRequest = PutRuleRequest.builder()
            .description("Created by using the AWS SDK for Java v2")
            .name(eventRuleName)
            .eventPattern(pattern)
            .roleArn(roleArn)
            .build();

        PutRuleResponse ruleResponse = eventBrClient.putRule(ruleRequest);
        System.out.println("The ARN of the new rule is " +
ruleResponse.ruleArn());

    } catch (EventBridgeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

// Determine if the S3 bucket exists.
public static Boolean checkBucket(S3Client s3Client, String bucketName) {
    try {
        HeadBucketRequest headBucketRequest = HeadBucketRequest.builder()
            .bucket(bucketName)
            .build();

        s3Client.headBucket(headBucketRequest);
        return true;
    }
}

```

```
    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }
    return false;
}

// Set the S3 bucket notification configuration.
public static void setBucketNotification(S3Client s3Client, String bucketName) {
    try {
        EventBridgeConfiguration eventBridgeConfiguration =
EventBridgeConfiguration.builder()
            .build();

        NotificationConfiguration configuration =
NotificationConfiguration.builder()
            .eventBridgeConfiguration(eventBridgeConfiguration)
            .build();

        PutBucketNotificationConfigurationRequest configurationRequest =
PutBucketNotificationConfigurationRequest
            .builder()
            .bucket(bucketName)
            .notificationConfiguration(configuration)
            .skipDestinationValidation(true)
            .build();

        s3Client.putBucketNotificationConfiguration(configurationRequest);
        System.out.println("Added bucket " + bucketName + " with EventBridge
events enabled.");
    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void createBucket(S3Client s3Client, String bucketName) {
    try {
        S3Waiter s3Waiter = s3Client.waiter();
        CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
            .bucket(bucketName)
            .build();

        s3Client.createBucket(bucketRequest);
    }
}
```

```
        HeadBucketRequest bucketRequestWait = HeadBucketRequest.builder()
            .bucket(bucketName)
            .build();

        // Wait until the bucket is created and print out the response.
        WaiterResponse<HeadBucketResponse> waiterResponse =
s3Waiter.waitUntilBucketExists(bucketRequestWait);
        waiterResponse.matched().response().ifPresent(System.out::println);
        System.out.println(bucketName + " is ready");

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static String createIAMRole(IamClient iam, String rolename, String
polJSON) {
    try {
        CreateRoleRequest request = CreateRoleRequest.builder()
            .roleName(rolename)
            .assumeRolePolicyDocument(polJSON)
            .description("Created using the AWS SDK for Java")
            .build();

        CreateRoleResponse response = iam.createRole(request);
        AttachRolePolicyRequest rolePolicyRequest =
AttachRolePolicyRequest.builder()
            .roleName(rolename)
            .policyArn("arn:aws:iam::aws:policy/
AmazonEventBridgeFullAccess")
            .build();

        iam.attachRolePolicy(rolePolicyRequest);
        return response.role().arn();

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
}
```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella Documentazione di riferimento delle API AWS SDK for Java 2.x .
  - [DeleteRule](#)
  - [DescribeRule](#)
  - [DisableRule](#)
  - [EnableRule](#)
  - [ListRuleNamesByTarget](#)
  - [ListRules](#)
  - [ListTargetsByRule](#)
  - [PutEvents](#)
  - [PutRule](#)
  - [PutTargets](#)

## Esempi di previsione con SDK for Java 2.x

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando AWS SDK for Java 2.x with Forecast.

Le operazioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le operazioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati e negli esempi tra servizi.

Scenari: esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio.

Ogni esempio include un collegamento a GitHub, dove è possibile trovare istruzioni su come configurare ed eseguire il codice nel contesto.

### Argomenti

- [Azioni](#)

## Azioni

### CreateDataset

Il seguente esempio di codice mostra come utilizzare `CreateDataset`.

SDK per Java 2.x

#### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.forecast.ForecastClient;
import software.amazon.awssdk.services.forecast.model.CreateDatasetRequest;
import software.amazon.awssdk.services.forecast.model.Schema;
import software.amazon.awssdk.services.forecast.model.SchemaAttribute;
import software.amazon.awssdk.services.forecast.model.CreateDatasetResponse;
import software.amazon.awssdk.services.forecast.model.ForecastException;
import java.util.ArrayList;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateDataSet {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <name>\s

                Where:
                name - The name of the data set.\s
        """;
```

```
        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String name = args[0];
        Region region = Region.US_WEST_2;
        ForecastClient forecast = ForecastClient.builder()
            .region(region)
            .build();

        String myDataSetARN = createForecastDataSet(forecast, name);
        System.out.println("The ARN of the new data set is " + myDataSetARN);
        forecast.close();
    }

    public static String createForecastDataSet(ForecastClient forecast, String name)
    {
        try {
            Schema schema = Schema.builder()
                .attributes(getSchema())
                .build();

            CreateDatasetRequest datasetRequest = CreateDatasetRequest.builder()
                .datasetName(name)
                .domain("CUSTOM")
                .datasetType("RELATED_TIME_SERIES")
                .dataFrequency("D")
                .schema(schema)
                .build();

            CreateDatasetResponse response = forecast.createDataset(datasetRequest);
            return response.datasetArn();

        } catch (ForecastException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }

        return "";
    }

    // Create a SchemaAttribute list required to create a data set.
    private static List<SchemaAttribute> getSchema() {
```



```
List<SchemaAttribute> schemaList = new ArrayList<>();
SchemaAttribute att1 = SchemaAttribute.builder()
    .attributeName("item_id")
    .attributeType("string")
    .build();

SchemaAttribute att2 = SchemaAttribute.builder()
    .attributeName("timestamp")
    .attributeType("timestamp")
    .build();

SchemaAttribute att3 = SchemaAttribute.builder()
    .attributeName("target_value")
    .attributeType("float")
    .build();

// Push the SchemaAttribute objects to the List.
schemaList.add(att1);
schemaList.add(att2);
schemaList.add(att3);
return schemaList;
}
}
```

- Per i dettagli sull'API, [CreateDataset](#) consulta AWS SDK for Java 2.x API Reference.

## CreateForecast

Il seguente esempio di codice mostra come utilizzare `CreateForecast`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.forecast.ForecastClient;
```

```
import software.amazon.awssdk.services.forecast.model.CreateForecastRequest;
import software.amazon.awssdk.services.forecast.model.CreateForecastResponse;
import software.amazon.awssdk.services.forecast.model.ForecastException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateForecast {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <name> <predictorArn>\s

            Where:
                name - The name of the forecast.\s
                predictorArn - The arn of the predictor to use.\s

            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String name = args[0];
        String predictorArn = args[1];
        Region region = Region.US_WEST_2;
        ForecastClient forecast = ForecastClient.builder()
            .region(region)
            .build();

        String forecastArn = createNewForecast(forecast, name, predictorArn);
        System.out.println("The ARN of the new forecast is " + forecastArn);
        forecast.close();
    }

    public static String createNewForecast(ForecastClient forecast, String name,
        String predictorArn) {
```

```
    try {
        CreateForecastRequest forecastRequest = CreateForecastRequest.builder()
            .forecastName(name)
            .predictorArn(predictorArn)
            .build();

        CreateForecastResponse response =
forecast.createForecast(forecastRequest);
        return response.forecastArn();

    } catch (ForecastException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
}
```

- Per i dettagli sull'API, [CreateForecast](#) consulta AWS SDK for Java 2.x API Reference.

## DeleteDataset

Il seguente esempio di codice mostra come utilizzare `DeleteDataset`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.forecast.ForecastClient;
import software.amazon.awssdk.services.forecast.model.DeleteDatasetRequest;
import software.amazon.awssdk.services.forecast.model.ForecastException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 */
```

```
* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class DeleteDataset {

    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <datasetARN>\s

            Where:
                datasetARN - The ARN of the data set to delete.\s
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String datasetARN = args[0];
        Region region = Region.US_WEST_2;
        ForecastClient forecast = ForecastClient.builder()
            .region(region)
            .build();

        deleteForecastDataSet(forecast, datasetARN);
        forecast.close();
    }

    public static void deleteForecastDataSet(ForecastClient forecast, String
myDataSetARN) {
        try {
            DeleteDatasetRequest deleteRequest = DeleteDatasetRequest.builder()
                .datasetArn(myDataSetARN)
                .build();

            forecast.deleteDataset(deleteRequest);
            System.out.println("The Data Set was deleted");

        } catch (ForecastException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

```
    }  
  }  
}
```

- Per i dettagli sull'API, [DeleteDataset](#) consulta AWS SDK for Java 2.x API Reference.

## DeleteForecast

Il seguente esempio di codice mostra come utilizzare `DeleteForecast`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;  
import software.amazon.awssdk.services.forecast.ForecastClient;  
import software.amazon.awssdk.services.forecast.model.DeleteDatasetRequest;  
import software.amazon.awssdk.services.forecast.model.ForecastException;  
  
/**  
 * Before running this Java V2 code example, set up your development  
 * environment, including your credentials.  
 *  
 * For more information, see the following documentation topic:  
 *  
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html  
 */  
public class DeleteDataset {  
  
    public static void main(String[] args) {  
        final String usage = ""  
  
            Usage:  
            <datasetARN>\s  
  
            Where:
```

```

        datasetARN - The ARN of the data set to delete.\s
        """;

    if (args.length != 1) {
        System.out.println(usage);
        System.exit(1);
    }

    String datasetARN = args[0];
    Region region = Region.US_WEST_2;
    ForecastClient forecast = ForecastClient.builder()
        .region(region)
        .build();

    deleteForecastDataSet(forecast, datasetARN);
    forecast.close();
}

public static void deleteForecastDataSet(ForecastClient forecast, String
myDataSetARN) {
    try {
        DeleteDatasetRequest deleteRequest = DeleteDatasetRequest.builder()
            .datasetArn(myDataSetARN)
            .build();

        forecast.deleteDataset(deleteRequest);
        System.out.println("The Data Set was deleted");

    } catch (ForecastException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
}

```

- Per i dettagli sull'API, [DeleteForecast](#) consulta AWS SDK for Java 2.x API Reference.

## DescribeForecast

Il seguente esempio di codice mostra come utilizzare `DescribeForecast`.

## SDK per Java 2.x

 Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.forecast.ForecastClient;
import software.amazon.awssdk.services.forecast.model.DescribeForecastRequest;
import software.amazon.awssdk.services.forecast.model.DescribeForecastResponse;
import software.amazon.awssdk.services.forecast.model.ForecastException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DescribeForecast {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <forecastarn>\s

                Where:
                forecastarn - The arn of the forecast (for example,
                "arn:aws:forecast:us-west-2:xxxxx322:forecast/my_forecast)
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String forecastarn = args[0];
        Region region = Region.US_WEST_2;
        ForecastClient forecast = ForecastClient.builder()
```

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

    describe(forecast, forecastarn);
    forecast.close();
}

public static void describe(ForecastClient forecast, String forecastarn) {
    try {
        DescribeForecastRequest request = DescribeForecastRequest.builder()
            .forecastArn(forecastarn)
            .build();

        DescribeForecastResponse response = forecast.describeForecast(request);
        System.out.println("The name of the forecast is " +
response.forecastName());

    } catch (ForecastException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Per i dettagli sull'API, [DescribeForecast](#) consulta AWS SDK for Java 2.x API Reference.

## ListDatasetGroups

Il seguente esempio di codice mostra come utilizzare `ListDatasetGroups`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.forecast.ForecastClient;
import software.amazon.awssdk.services.forecast.model.DatasetGroupSummary;
```



```
import software.amazon.awssdk.services.forecast.model.ListDatasetGroupsRequest;
import software.amazon.awssdk.services.forecast.model.ListDatasetGroupsResponse;
import software.amazon.awssdk.services.forecast.model.ForecastException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListDataSetGroups {
    public static void main(String[] args) {
        Region region = Region.US_WEST_2;
        ForecastClient forecast = ForecastClient.builder()
            .region(region)
            .build();

        listDataGroups(forecast);
        forecast.close();
    }

    public static void listDataGroups(ForecastClient forecast) {
        try {
            ListDatasetGroupsRequest group = ListDatasetGroupsRequest.builder()
                .maxResults(10)
                .build();

            ListDatasetGroupsResponse response = forecast.listDatasetGroups(group);
            List<DatasetGroupSummary> groups = response.datasetGroups();
            for (DatasetGroupSummary myGroup : groups) {
                System.out.println("The Data Set name is " +
myGroup.datasetGroupName());
            }

        } catch (ForecastException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Per i dettagli sull'API, [ListDatasetGroups](#) consulta AWS SDK for Java 2.x API Reference.

## ListForecasts

Il seguente esempio di codice mostra come utilizzare `ListForecasts`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.forecast.ForecastClient;
import software.amazon.awssdk.services.forecast.model.ListForecastsResponse;
import software.amazon.awssdk.services.forecast.model.ListForecastsRequest;
import software.amazon.awssdk.services.forecast.model.ForecastSummary;
import software.amazon.awssdk.services.forecast.model.ForecastException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListForecasts {

    public static void main(String[] args) {
        Region region = Region.US_WEST_2;
        ForecastClient forecast = ForecastClient.builder()
            .region(region)
            .build();

        listAllForecasts(forecast);
        forecast.close();
    }
}
```

```
    }

    public static void listAllForeCasts(ForecastClient forecast) {
        try {
            ListForecastsRequest request = ListForecastsRequest.builder()
                .maxResults(10)
                .build();

            ListForecastsResponse response = forecast.listForecasts(request);
            List<ForecastSummary> forecasts = response.forecasts();
            for (ForecastSummary forecastSummary : forecasts) {
                System.out.println("The name of the forecast is " +
forecastSummary.forecastName());
            }

        } catch (ForecastException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Per i dettagli sull'API, [ListForecasts](#) consulta AWS SDK for Java 2.x API Reference.

## AWS Glue esempi che utilizzano SDK for Java 2.x

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando AWS SDK for Java 2.x with AWS Glue.

Le operazioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le operazioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati e negli esempi tra servizi.

Scenari: esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio.

Ogni esempio include un collegamento a GitHub, dove è possibile trovare istruzioni su come configurare ed eseguire il codice nel contesto.

### Nozioni di base

## Salve AWS Glue

L'esempio di codice seguente mostra come iniziare a utilizzare AWS Glue.

### SDK per Java 2.x

#### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
package com.example.glue;

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.glue.GlueClient;
import software.amazon.awssdk.services.glue.model.ListJobsRequest;
import software.amazon.awssdk.services.glue.model.ListJobsResponse;
import java.util.List;

public class HelloGlue {
    public static void main(String[] args) {
        GlueClient glueClient = GlueClient.builder()
            .region(Region.US_EAST_1)
            .build();

        listJobs(glueClient);
    }

    public static void listJobs(GlueClient glueClient) {
        ListJobsRequest request = ListJobsRequest.builder()
            .maxResults(10)
            .build();
        ListJobsResponse response = glueClient.listJobs(request);
        List<String> jobList = response.jobNames();
        jobList.forEach(job -> {
            System.out.println("Job Name: " + job);
        });
    }
}
```

- Per i dettagli sull'API, [ListJobs](#) consulta AWS SDK for Java 2.x API Reference.

## Argomenti

- [Azioni](#)
- [Scenari](#)

## Azioni

### CreateCrawler

Il seguente esempio di codice mostra come utilizzare `CreateCrawler`.

#### SDK per Java 2.x

#### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.glue.GlueClient;
import software.amazon.awssdk.services.glue.model.CreateCrawlerRequest;
import software.amazon.awssdk.services.glue.model.CrawlerTargets;
import software.amazon.awssdk.services.glue.model.GlueException;
import software.amazon.awssdk.services.glue.model.S3Target;
import java.util.ArrayList;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateCrawler {
    public static void main(String[] args) {
        final String usage = ""
```

```

Usage:
    <IAM> <s3Path> <cron> <dbName> <crawlerName>

Where:
    IAM - The ARN of the IAM role that has AWS Glue and S3
permissions.\s
    s3Path - The Amazon Simple Storage Service (Amazon S3) target
that contains data (for example, CSV data).
    cron - A cron expression used to specify the schedule (i.e.,
cron(15 12 * * ? *).
    dbName - The database name.\s
    crawlerName - The name of the crawler.\s
    """;

if (args.length != 5) {
    System.out.println(usage);
    System.exit(1);
}

String iam = args[0];
String s3Path = args[1];
String cron = args[2];
String dbName = args[3];
String crawlerName = args[4];
Region region = Region.US_EAST_1;
GlueClient glueClient = GlueClient.builder()
    .region(region)
    .build();

createGlueCrawler(glueClient, iam, s3Path, cron, dbName, crawlerName);
glueClient.close();
}

public static void createGlueCrawler(GlueClient glueClient,
    String iam,
    String s3Path,
    String cron,
    String dbName,
    String crawlerName) {

    try {
        S3Target s3Target = S3Target.builder()
            .path(s3Path)

```

```
        .build();

// Add the S3Target to a list.
List<S3Target> targetList = new ArrayList<>();
targetList.add(s3Target);

CrawlerTargets targets = CrawlerTargets.builder()
    .s3Targets(targetList)
    .build();

CreateCrawlerRequest crawlerRequest = CreateCrawlerRequest.builder()
    .databaseName(dbName)
    .name(crawlerName)
    .description("Created by the AWS Glue Java API")
    .targets(targets)
    .role(iam)
    .schedule(cron)
    .build();

glueClient.createCrawler(crawlerRequest);
System.out.println(crawlerName + " was successfully created");

} catch (GlueException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

- Per i dettagli sull'API, [CreateCrawler](#) consulta AWS SDK for Java 2.x API Reference.

## GetCrawler

Il seguente esempio di codice mostra come utilizzare `GetCrawler`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.glue.GlueClient;
import software.amazon.awssdk.services.glue.model.GetCrawlerRequest;
import software.amazon.awssdk.services.glue.model.GetCrawlerResponse;
import software.amazon.awssdk.services.glue.model.GlueException;
import java.time.Instant;
import java.time.ZoneId;
import java.time.format.DateTimeFormatter;
import java.time.format.FormatStyle;
import java.util.Locale;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class GetCrawler {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <crawlerName>

                Where:
                crawlerName - The name of the crawler.\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String crawlerName = args[0];
        Region region = Region.US_EAST_1;
        GlueClient glueClient = GlueClient.builder()
            .region(region)
            .build();

        getSpecificCrawler(glueClient, crawlerName);
        glueClient.close();
    }
}
```



```
}

public static void getSpecificCrawler(GlueClient glueClient, String crawlerName)
{
    try {
        GetCrawlerRequest crawlerRequest = GetCrawlerRequest.builder()
            .name(crawlerName)
            .build();

        GetCrawlerResponse response = glueClient.getCrawler(crawlerRequest);
        Instant createDate = response.crawler().creationTime();

        // Convert the Instant to readable date
        DateTimeFormatter formatter =
        DateTimeFormatter.ofLocalizedDateTime(FormatStyle.SHORT)
            .withLocale(Locale.US)
            .withZone(ZoneId.systemDefault());

        formatter.format(createDate);
        System.out.println("The create date of the Crawler is " + createDate);

    } catch (GlueException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Per i dettagli sull'API, [GetCrawler](#) consulta AWS SDK for Java 2.x API Reference.

## GetDatabase

Il seguente esempio di codice mostra come utilizzare `GetDatabase`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.glue.GlueClient;
import software.amazon.awssdk.services.glue.model.GetDatabaseRequest;
import software.amazon.awssdk.services.glue.model.GetDatabaseResponse;
import software.amazon.awssdk.services.glue.model.GlueException;
import java.time.Instant;
import java.time.ZoneId;
import java.time.format.DateTimeFormatter;
import java.time.format.FormatStyle;
import java.util.Locale;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class GetDatabase {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <databaseName>

                Where:
                databaseName - The name of the database.\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String databaseName = args[0];
        Region region = Region.US_EAST_1;
        GlueClient glueClient = GlueClient.builder()
            .region(region)
            .build();

        getSpecificDatabase(glueClient, databaseName);
        glueClient.close();
    }
}
```

```
}

    public static void getSpecificDatabase(GlueClient glueClient, String
databaseName) {
        try {
            GetDatabaseRequest databasesRequest = GetDatabaseRequest.builder()
                .name(databaseName)
                .build();

            GetDatabaseResponse response = glueClient.getDatabase(databasesRequest);
            Instant createDate = response.database().createTime();

            // Convert the Instant to readable date.
            DateTimeFormatter formatter =
            DateTimeFormatter.ofLocalizedDateTime(FormatStyle.SHORT)
                .withLocale(Locale.US)
                .withZone(ZoneId.systemDefault());

            formatter.format(createDate);
            System.out.println("The create date of the database is " + createDate);

        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Per i dettagli sull'API, [GetDatabase](#) consulta AWS SDK for Java 2.x API Reference.

## GetTables

Il seguente esempio di codice mostra come utilizzare `GetTables`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.glue.GlueClient;
import software.amazon.awssdk.services.glue.model.GetTableRequest;
import software.amazon.awssdk.services.glue.model.GetTableResponse;
import software.amazon.awssdk.services.glue.model.GlueException;
import java.time.Instant;
import java.time.ZoneId;
import java.time.format.DateTimeFormatter;
import java.time.format.FormatStyle;
import java.util.Locale;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class GetTable {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <dbName> <tableName>

            Where:
                dbName - The database name.\s
                tableName - The name of the table.\s
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String dbName = args[0];
        String tableName = args[1];
        Region region = Region.US_EAST_1;
        GlueClient glueClient = GlueClient.builder()
            .region(region)
            .build();
    }
}
```

```
        getGlueTable(glueClient, dbName, tableName);
        glueClient.close();
    }

    public static void getGlueTable(GlueClient glueClient, String dbName, String
tableName) {
        try {
            GetTableRequest tableRequest = GetTableRequest.builder()
                .databaseName(dbName)
                .name(tableName)
                .build();

            GetTableResponse tableResponse = glueClient.getTable(tableRequest);
            Instant createDate = tableResponse.table().createTime();

            // Convert the Instant to readable date.
            DateTimeFormatter formatter =
            DateTimeFormatter.ofLocalizedDateTime(FormatStyle.SHORT)
                .withLocale(Locale.US)
                .withZone(ZoneId.systemDefault());

            formatter.format(createDate);
            System.out.println("The create date of the table is " + createDate);

        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Per i dettagli sull'API, [GetTables](#) consulta AWS SDK for Java 2.x API Reference.

## StartCrawler

Il seguente esempio di codice mostra come utilizzare `StartCrawler`.

## SDK per Java 2.x

 Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.glue.GlueClient;
import software.amazon.awssdk.services.glue.model.GlueException;
import software.amazon.awssdk.services.glue.model.StartCrawlerRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class StartCrawler {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <crawlerName>

                Where:
                crawlerName - The name of the crawler.\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String crawlerName = args[0];
        Region region = Region.US_EAST_1;
        GlueClient glueClient = GlueClient.builder()
                .region(region)
                .build();
```

```
        startSpecificCrawler(glueClient, crawlerName);
        glueClient.close();
    }

    public static void startSpecificCrawler(GlueClient glueClient, String
crawlerName) {
        try {
            StartCrawlerRequest crawlerRequest = StartCrawlerRequest.builder()
                .name(crawlerName)
                .build();

            glueClient.startCrawler(crawlerRequest);

        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Per i dettagli sull'API, [StartCrawler](#) consulta AWS SDK for Java 2.x API Reference.

## Scenari


### Nozioni di base su crawler e processi

L'esempio di codice seguente mostra come:

- Crea un crawler che esegue la scansione di un bucket Amazon S3 pubblico e genera un database di metadati in formato CSV.
- Elenca le informazioni su database e tabelle nel tuo AWS Glue Data Catalog.
- Crea un processo per estrarre i dati CSV dal bucket S3, trasformare i dati e caricare l'output in formato JSON in un altro bucket S3.
- Elenca le informazioni sulle esecuzioni dei processi, visualizza i dati trasformati e pulisci le risorse.

Per ulteriori informazioni, consulta [Tutorial: Guida introduttiva a AWS Glue Studio](#).

## SDK per Java 2.x

 Note

C'è altro da sapere GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
/**
 *
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * To set up the resources, see this documentation topic:
 *
 * https://docs.aws.amazon.com/glue/latest/ug/tutorial-add-crawler.html
 *
 * This example performs the following tasks:
 *
 * 1. Create a database.
 * 2. Create a crawler.
 * 3. Get a crawler.
 * 4. Start a crawler.
 * 5. Get a database.
 * 6. Get tables.
 * 7. Create a job.
 * 8. Start a job run.
 * 9. List all jobs.
 * 10. Get job runs.
 * 11. Delete a job.
 * 12. Delete a database.
 * 13. Delete a crawler.
 */

public class GlueScenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    public static void main(String[] args) throws InterruptedException {
```



```

final String usage = ""

    Usage:
        <iam> <s3Path> <cron> <dbName> <crawlerName> <jobName>\s

    Where:
        iam - The ARN of the IAM role that has AWS Glue and S3
permissions.\s
        s3Path - The Amazon Simple Storage Service (Amazon S3) target
that contains data (for example, CSV data).
        cron - A cron expression used to specify the schedule (i.e.,
cron(15 12 * * ? *).
        dbName - The database name.\s
        crawlerName - The name of the crawler.\s
        jobName - The name you assign to this job definition.
        scriptLocation - The Amazon S3 path to a script that runs a job.
        locationUri - The location of the database
        bucketNameSc - The Amazon S3 bucket name used when creating a
job

        """;

if (args.length != 9) {
    System.out.println(usage);
    System.exit(1);
}

String iam = args[0];
String s3Path = args[1];
String cron = args[2];
String dbName = args[3];
String crawlerName = args[4];
String jobName = args[5];
String scriptLocation = args[6];
String locationUri = args[7];
String bucketNameSc = args[8];

Region region = Region.US_EAST_1;
GlueClient glueClient = GlueClient.builder()
    .region(region)
    .build();
System.out.println(DASHES);
System.out.println("Welcome to the AWS Glue scenario.");
System.out.println(DASHES);

```

```
System.out.println(DASHES);
System.out.println("1. Create a database.");
createDatabase(glueClient, dbName, locationUri);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("2. Create a crawler.");
createGlueCrawler(glueClient, iam, s3Path, cron, dbName, crawlerName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("3. Get a crawler.");
getSpecificCrawler(glueClient, crawlerName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("4. Start a crawler.");
startSpecificCrawler(glueClient, crawlerName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("5. Get a database.");
getSpecificDatabase(glueClient, dbName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("*** Wait 5 min for the tables to become available");
TimeUnit.MINUTES.sleep(5);
System.out.println("6. Get tables.");
String myTableName = getGlueTables(glueClient, dbName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("7. Create a job.");
createJob(glueClient, jobName, iam, scriptLocation);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("8. Start a Job run.");
startJob(glueClient, jobName, dbName, myTableName, bucketNameSc);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("9. List all jobs.");
```

```
    getAllJobs(glueClient);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("10. Get job runs.");
    getJobRuns(glueClient, jobName);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("11. Delete a job.");
    deleteJob(glueClient, jobName);
    System.out.println("*** Wait 5 MIN for the " + crawlerName + " to stop");
    TimeUnit.MINUTES.sleep(5);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("12. Delete a database.");
    deleteDatabase(glueClient, dbName);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("Delete a crawler.");
    deleteSpecificCrawler(glueClient, crawlerName);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("Successfully completed the AWS Glue Scenario");
    System.out.println(DASHES);
}

public static void createDatabase(GlueClient glueClient, String dbName, String
locationUri) {
    try {
        DatabaseInput input = DatabaseInput.builder()
            .description("Built with the AWS SDK for Java V2")
            .name(dbName)
            .locationUri(locationUri)
            .build();

        CreateDatabaseRequest request = CreateDatabaseRequest.builder()
            .databaseInput(input)
            .build();

        glueClient.createDatabase(request);
    }
}
```

```
        System.out.println(dbName + " was successfully created");

    } catch (GlueException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void createGlueCrawler(GlueClient glueClient,
    String iam,
    String s3Path,
    String cron,
    String dbName,
    String crawlerName) {

    try {
        S3Target s3Target = S3Target.builder()
            .path(s3Path)
            .build();

        List<S3Target> targetList = new ArrayList<>();
        targetList.add(s3Target);
        CrawlerTargets targets = CrawlerTargets.builder()
            .s3Targets(targetList)
            .build();

        CreateCrawlerRequest crawlerRequest = CreateCrawlerRequest.builder()
            .databaseName(dbName)
            .name(crawlerName)
            .description("Created by the AWS Glue Java API")
            .targets(targets)
            .role(iam)
            .schedule(cron)
            .build();

        glueClient.createCrawler(crawlerRequest);
        System.out.println(crawlerName + " was successfully created");

    } catch (GlueException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

```
public static void getSpecificCrawler(GlueClient glueClient, String crawlerName)
{
    try {
        GetCrawlerRequest crawlerRequest = GetCrawlerRequest.builder()
            .name(crawlerName)
            .build();

        boolean ready = false;
        while (!ready) {
            GetCrawlerResponse response = glueClient.getCrawler(crawlerRequest);
            String status = response.crawler().stateAsString();
            if (status.compareTo("READY") == 0) {
                ready = true;
            }
            Thread.sleep(3000);
        }

        System.out.println("The crawler is now ready");

    } catch (GlueException | InterruptedException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void startSpecificCrawler(GlueClient glueClient, String
crawlerName) {
    try {
        StartCrawlerRequest crawlerRequest = StartCrawlerRequest.builder()
            .name(crawlerName)
            .build();

        glueClient.startCrawler(crawlerRequest);
        System.out.println(crawlerName + " was successfully started!");

    } catch (GlueException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void getSpecificDatabase(GlueClient glueClient, String
databaseName) {
    try {
```

```
        GetDatabaseRequest databasesRequest = GetDatabaseRequest.builder()
            .name(databaseName)
            .build();

        GetDatabaseResponse response = glueClient.getDatabase(databasesRequest);
        Instant createDate = response.database().createTime();

        // Convert the Instant to readable date.
        DateTimeFormatter formatter =
        DateTimeFormatter.ofLocalizedDateTime(FormatStyle.SHORT)
            .withLocale(Locale.US)
            .withZone(ZoneId.systemDefault());

        formatter.format(createDate);
        System.out.println("The create date of the database is " + createDate);

    } catch (GlueException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static String getGlueTables(GlueClient glueClient, String dbName) {
    String myTableName = "";
    try {
        GetTablesRequest tableRequest = GetTablesRequest.builder()
            .databaseName(dbName)
            .build();

        GetTablesResponse response = glueClient.getTables(tableRequest);
        List<Table> tables = response.tableList();
        if (tables.isEmpty()) {
            System.out.println("No tables were returned");
        } else {
            for (Table table : tables) {
                myTableName = table.name();
                System.out.println("Table name is: " + myTableName);
            }
        }
    } catch (GlueException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

```
        return myTableName;
    }

    public static void startJob(GlueClient glueClient, String jobName, String
inputDatabase, String inputTable,
        String outBucket) {
    try {
        Map<String, String> myMap = new HashMap<>();
        myMap.put("--input_database", inputDatabase);
        myMap.put("--input_table", inputTable);
        myMap.put("--output_bucket_url", outBucket);

        StartJobRunRequest runRequest = StartJobRunRequest.builder()
            .workerType(WorkerType.G_1_X)
            .numberOfWorkers(10)
            .arguments(myMap)
            .jobName(jobName)
            .build();

        StartJobRunResponse response = glueClient.startJobRun(runRequest);
        System.out.println("The request Id of the job is " +
response.responseMetadata().requestId());

    } catch (GlueException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

    public static void createJob(GlueClient glueClient, String jobName, String iam,
String scriptLocation) {
    try {
        JobCommand command = JobCommand.builder()
            .pythonVersion("3")
            .name("glueetl")
            .scriptLocation(scriptLocation)
            .build();

        CreateJobRequest jobRequest = CreateJobRequest.builder()
            .description("A Job created by using the AWS SDK for Java V2")
            .glueVersion("2.0")
            .workerType(WorkerType.G_1_X)
            .numberOfWorkers(10)
            .name(jobName)
```

```
        .role(iam)
        .command(command)
        .build();

    glueClient.createJob(jobRequest);
    System.out.println(jobName + " was successfully created.");

} catch (GlueException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}

public static void getAllJobs(GlueClient glueClient) {
    try {
        GetJobsRequest jobsRequest = GetJobsRequest.builder()
            .maxResults(10)
            .build();

        GetJobsResponse jobsResponse = glueClient.getJobs(jobsRequest);
        List<Job> jobs = jobsResponse.jobs();
        for (Job job : jobs) {
            System.out.println("Job name is : " + job.name());
            System.out.println("The job worker type is : " +
job.workerType().name());
        }

    } catch (GlueException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void getJobRuns(GlueClient glueClient, String jobName) {
    try {
        GetJobRunsRequest runsRequest = GetJobRunsRequest.builder()
            .jobName(jobName)
            .maxResults(20)
            .build();

        boolean jobDone = false;
        while (!jobDone) {
            GetJobRunsResponse response = glueClient.getJobRuns(runsRequest);
            List<JobRun> jobRuns = response.jobRuns();
```



```
        for (JobRun jobRun : jobRuns) {
            String jobState = jobRun.jobRunState().name();
            if (jobState.compareTo("SUCCEEDED") == 0) {
                System.out.println(jobName + " has succeeded");
                jobDone = true;

            } else if (jobState.compareTo("STOPPED") == 0) {
                System.out.println("Job run has stopped");
                jobDone = true;

            } else if (jobState.compareTo("FAILED") == 0) {
                System.out.println("Job run has failed");
                jobDone = true;

            } else if (jobState.compareTo("TIMEOUT") == 0) {
                System.out.println("Job run has timed out");
                jobDone = true;

            } else {
                System.out.println("*** Job run state is " +
jobRun.jobRunState().name());
                System.out.println("Job run Id is " + jobRun.id());
                System.out.println("The Glue version is " +
jobRun.glueVersion());
            }
            TimeUnit.SECONDS.sleep(5);
        }

    } catch (GlueException | InterruptedException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void deleteJob(GlueClient glueClient, String jobName) {
    try {
        DeleteJobRequest jobRequest = DeleteJobRequest.builder()
            .jobName(jobName)
            .build();

        glueClient.deleteJob(jobRequest);
        System.out.println(jobName + " was successfully deleted");
    }
}
```

```
        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void deleteDatabase(GlueClient glueClient, String databaseName) {
        try {
            DeleteDatabaseRequest request = DeleteDatabaseRequest.builder()
                .name(databaseName)
                .build();

            glueClient.deleteDatabase(request);
            System.out.println(databaseName + " was successfully deleted");

        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void deleteSpecificCrawler(GlueClient glueClient, String
crawlerName) {
        try {
            DeleteCrawlerRequest deleteCrawlerRequest =
DeleteCrawlerRequest.builder()
                .name(crawlerName)
                .build();

            glueClient.deleteCrawler(deleteCrawlerRequest);
            System.out.println(crawlerName + " was deleted");

        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella Documentazione di riferimento delle API AWS SDK for Java 2.x .
  - [CreateCrawler](#)

- [CreateJob](#)
- [DeleteCrawler](#)
- [DeleteDatabase](#)
- [DeleteJob](#)
- [DeleteTable](#)
- [GetCrawler](#)
- [GetDatabase](#)
- [GetDatabases](#)
- [GetJob](#)
- [GetJobRun](#)
- [GetJobRuns](#)
- [GetTables](#)
- [ListJobs](#)
- [StartCrawler](#)
- [StartJobRun](#)

## HealthImaging esempi che utilizzano SDK for Java 2.x

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando AWS SDK for Java 2.x with HealthImaging.

Le operazioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le operazioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati e negli esempi tra servizi.

Scenari: esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio.

Ogni esempio include un collegamento a GitHub, dove è possibile trovare istruzioni su come configurare ed eseguire il codice nel contesto.

### Argomenti

- [Azioni](#)
- [Scenari](#)

## Azioni

### CopyImageSet

Il seguente esempio di codice mostra come utilizzare `CopyImageSet`.

SDK per Java 2.x

```
public static String copyMedicalImageSet(MedicalImagingClient
medicalImagingClient,
    String datastoreId,
    String imageSetId,
    String latestVersionId,
    String destinationImageSetId,
    String destinationVersionId) {

    try {
        CopySourceImageSetInformation copySourceImageSetInformation =
CopySourceImageSetInformation.builder()
            .latestVersionId(latestVersionId)
            .build();

        CopyImageSetInformation.Builder copyImageSetBuilder =
CopyImageSetInformation.builder()
            .sourceImageSet(copySourceImageSetInformation);

        if (destinationImageSetId != null) {
            copyImageSetBuilder =
copyImageSetBuilder.destinationImageSet(CopyDestinationImageSet.builder()
                .imageSetId(destinationImageSetId)
                .latestVersionId(destinationVersionId)
                .build());
        }

        CopyImageSetRequest copyImageSetRequest = CopyImageSetRequest.builder()
            .datastoreId(datastoreId)
            .sourceImageSetId(imageSetId)
            .copyImageSetInformation(copyImageSetBuilder.build())
            .build();

        CopyImageSetResponse response =
medicalImagingClient.copyImageSet(copyImageSetRequest);

        return response.destinationImageSetProperties().imageSetId();
    }
}
```

```
    } catch (MedicalImagingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }

    return "";
}
```

- Per i dettagli sull'API, consulta la [CopyImageSet](#) sezione AWS SDK for Java 2.x API Reference.

### Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

## CreateDatastore


Il seguente esempio di codice mostra come usare `CreateDatastore`.

SDK per Java 2.x

```
public static String createMedicalImageDatastore(MedicalImagingClient
medicalImagingClient,
        String datastoreName) {
    try {
        CreateDatastoreRequest datastoreRequest =
CreateDatastoreRequest.builder()
            .datastoreName(datastoreName)
            .build();
        CreateDatastoreResponse response =
medicalImagingClient.createDatastore(datastoreRequest);
        return response.datastoreId();
    } catch (MedicalImagingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }

    return "";
}
```

- Per i dettagli sull'API, consulta la [CreateDatastore](#) sezione AWS SDK for Java 2.x API Reference.

 Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

## DeleteDatastore

Il seguente esempio di codice mostra come usare `DeleteDatastore`.

SDK per Java 2.x

```
public static void deleteMedicalImagingDatastore(MedicalImagingClient
medicalImagingClient,
    String datastoreID) {
    try {
        DeleteDatastoreRequest datastoreRequest =
DeleteDatastoreRequest.builder()
            .datastoreId(datastoreID)
            .build();
        medicalImagingClient.deleteDatastore(datastoreRequest);
    } catch (MedicalImagingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Per i dettagli sull'API, consulta la [DeleteDatastore](#) sezione AWS SDK for Java 2.x API Reference.

 Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

## DeleteImageSet

Il seguente esempio di codice mostra come usare `DeleteImageSet`.

SDK per Java 2.x

```
public static void deleteMedicalImageSet(MedicalImagingClient
medicalImagingClient,
    String datastoreId,
    String imagesetId) {
    try {
        DeleteImageSetRequest deleteImageSetRequest =
DeleteImageSetRequest.builder()
            .datastoreId(datastoreId)
            .imageSetId(imagesetId)
            .build();

        medicalImagingClient.deleteImageSet(deleteImageSetRequest);

        System.out.println("The image set was deleted.");
    } catch (MedicalImagingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Per i dettagli sull'API, consulta la [DeleteImageSet](#) sezione AWS SDK for Java 2.x API Reference.

### Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

## GetDICOMImportJob

Il seguente esempio di codice mostra come usare `GetDICOMImportJob`.

## SDK per Java 2.x

```
public static DICOMImportJobProperties getDicomImportJob(MedicalImagingClient
medicalImagingClient,
                String datastoreId,
                String jobId) {

    try {
        GetDicomImportJobRequest getDicomImportJobRequest =
        GetDicomImportJobRequest.builder()
            .datastoreId(datastoreId)
            .jobId(jobId)
            .build();
        GetDicomImportJobResponse response =
        medicalImagingClient.getDICOMImportJob(getDicomImportJobRequest);
        return response.jobProperties();
    } catch (MedicalImagingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }

    return null;
}
```

- Per i dettagli sull'API, consulta [getDICOM ImportJob](#) in AWS SDK for Java 2.x API Reference.

### Note

C'è di più su. [GitHub](#) Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

## GetDatastore

Il seguente esempio di codice mostra come usare `GetDatastore`.

## SDK per Java 2.x

```
public static DatastoreProperties getMedicalImageDatastore(MedicalImagingClient
medicalImagingClient,
                String datastoreID) {
```



```
    try {
        GetDatastoreRequest datastoreRequest = GetDatastoreRequest.builder()
            .datastoreId(datastoreId)
            .build();
        GetDatastoreResponse response =
medicalImagingClient.getDatastore(datastoreRequest);
        return response.datastoreProperties();
    } catch (MedicalImagingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }

    return null;
}
```

- Per i dettagli sull'API, consulta la [GetDatastore](#) sezione AWS SDK for Java 2.x API Reference.

#### Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

## GetImageFrame

Il seguente esempio di codice mostra come usare `GetImageFrame`.

SDK per Java 2.x

```
    public static void getMedicalImageSetFrame(MedicalImagingClient
medicalImagingClient,
        String destinationPath,
        String datastoreId,
        String imagesetId,
        String imageFrameId) {

        try {
            GetImageFrameRequest getImageSetMetadataRequest =
GetImageFrameRequest.builder()
                .datastoreId(datastoreId)
                .imageSetId(imagesetId)
```

```

        .imageFrameInformation(ImageFrameInformation.builder()
                                .imageFrameId(imageFrameId)
                                .build())
            .build();

medicalImagingClient.getImageFrame(getImageSetMetadataRequest,
    FileSystems.getDefault().getPath(destinationPath));

        System.out.println("Image frame downloaded to " +
destinationPath);
    } catch (MedicalImagingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

```

- Per i dettagli sull'API, consulta la [GetImageFrames](#) sezione AWS SDK for Java 2.x API Reference.

#### Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

## GetImageSet

Il seguente esempio di codice mostra come usare `GetImageSet`.

SDK per Java 2.x

```

    public static GetImageSetResponse getMedicalImageSet(MedicalImagingClient
medicalImagingClient,
        String datastoreId,
        String imagesetId,
        String versionId) {
    try {
        GetImageSetRequest.Builder getImageSetRequestBuilder =
GetImageSetRequest.builder()

```

```
        .datastoreId(datastoreId)
        .imageSetId(imagesetId);

    if (versionId != null) {
        getImageSetRequestBuilder =
getImageSetRequestBuilder.versionId(versionId);
    }

    return
medicalImagingClient.getImageSet(getImageSetRequestBuilder.build());
    } catch (MedicalImagingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }

    return null;
}
```

- Per i dettagli sull'API, consulta la [GetImageSet](#) sezione AWS SDK for Java 2.x API Reference.

#### Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

## GetImageSetMetadata

Il seguente esempio di codice mostra come usare `GetImageSetMetadata`.

SDK per Java 2.x

```
public static void getMedicalImageSetMetadata(MedicalImagingClient
medicalImagingClient,
    String destinationPath,
    String datastoreId,
    String imagesetId,
    String versionId) {

    try {
        GetImageSetMetadataRequest.Builder getImageSetMetadataRequestBuilder =
GetImageSetMetadataRequest.builder()
```

```

        .datastoreId(datastoreId)
        .imageSetId(imagesetId);

        if (versionId != null) {
            getImageSetMetadataRequestBuilder =
getImageSetMetadataRequestBuilder.versionId(versionId);
        }

medicalImagingClient.getImageSetMetadata(getImageSetMetadataRequestBuilder.build(),
        FileSystems.getDefault().getPath(destinationPath));

        System.out.println("Metadata downloaded to " + destinationPath);
    } catch (MedicalImagingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

```

- Per i dettagli sull'API, consulta la [GetImageSetMetadata](#) sezione AWS SDK for Java 2.x API Reference.

#### Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

## ListDICOMImportJobs

Il seguente esempio di codice mostra come usare `ListDICOMImportJobs`.

SDK per Java 2.x

```

public static List<DICOMImportJobSummary>
listDicomImportJobs(MedicalImagingClient medicalImagingClient,
        String datastoreId) {

    try {
        ListDicomImportJobsRequest listDicomImportJobsRequest =
ListDicomImportJobsRequest.builder()

```

```

        .datastoreId(datastoreId)
        .build();
    ListDicomImportJobsResponse response =
medicalImagingClient.listDICOMImportJobs(listDicomImportJobsRequest);
    return response.jobSummaries();
} catch (MedicalImagingException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}

return new ArrayList<>();
}

```

- Per i dettagli sull'API, consulta [ListDicom ImportJobs](#) in AWS SDK for Java 2.x API Reference.

### Note

C'è di più su. [GitHub](#) Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

## ListDatastores

Il seguente esempio di codice mostra come usare `ListDatastores`.

SDK per Java 2.x

```

public static List<DatastoreSummary>
listMedicalImagingDatastores(MedicalImagingClient medicalImagingClient) {
    try {
        ListDatastoresRequest datastoreRequest = ListDatastoresRequest.builder()
            .build();
        ListDatastoresIterable responses =
medicalImagingClient.listDatastoresPaginator(datastoreRequest);
        List<DatastoreSummary> datastoreSummaries = new ArrayList<>();

        responses.stream().forEach(response ->
datastoreSummaries.addAll(response.datastoreSummaries()));

        return datastoreSummaries;
    } catch (MedicalImagingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }
}

```

```
        System.exit(1);
    }

    return null;
}
```

- Per i dettagli sull'API, consulta la [ListDatastores](#) sezione AWS SDK for Java 2.x API Reference.

#### Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

## ListImageSetVersions

Il seguente esempio di codice mostra come usare `ListImageSetVersions`.

SDK per Java 2.x

```
public static List<ImageSetProperties>
listMedicalImageSetVersions(MedicalImagingClient medicalImagingClient,
    String datastoreId,
    String imagesetId) {
    try {
        ListImageSetVersionsRequest getImageSetRequest =
ListImageSetVersionsRequest.builder()
            .datastoreId(datastoreId)
            .imageSetId(imagesetId)
            .build();

        ListImageSetVersionsIterable responses = medicalImagingClient
            .listImageSetVersionsPaginator(getImageSetRequest);
        List<ImageSetProperties> imageSetProperties = new ArrayList<>();
        responses.stream().forEach(response ->
imageSetProperties.addAll(response.imageSetPropertiesList()));

        return imageSetProperties;
    } catch (MedicalImagingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

```
        return null;
    }
```

- Per i dettagli sull'API, consulta la [ListImageSetVersions](#) sezione AWS SDK for Java 2.x API Reference.

### Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

## ListTagsForResource

Il seguente esempio di codice mostra come usare `ListTagsForResource`.

SDK per Java 2.x

```
public static ListTagsForResourceResponse
listMedicalImagingResourceTags(MedicalImagingClient medicalImagingClient,
    String resourceArn) {
    try {
        ListTagsForResourceRequest listTagsForResourceRequest =
ListTagsForResourceRequest.builder()
            .resourceArn(resourceArn)
            .build();

        return
medicalImagingClient.listTagsForResource(listTagsForResourceRequest);
    } catch (MedicalImagingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }

    return null;
}
```

- Per i dettagli sull'API, consulta la [ListTagsForResource](#) sezione AWS SDK for Java 2.x API Reference.

**Note**

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

## SearchImageSets

Il seguente esempio di codice mostra come usare `SearchImageSets`.

SDK per Java 2.x

La funzione di utilità per la ricerca di set di immagini.

```
public static List<ImageSetsMetadataSummary> searchMedicalImagingImageSets(
    MedicalImagingClient medicalImagingClient,
    String datastoreId, SearchCriteria searchCriteria) {
    try {
        SearchImageSetsRequest datastoreRequest =
SearchImageSetsRequest.builder()
            .datastoreId(datastoreId)
            .searchCriteria(searchCriteria)
            .build();
        SearchImageSetsIterable responses = medicalImagingClient
            .searchImageSetsPaginator(datastoreRequest);
        List<ImageSetsMetadataSummary> imageSetsMetadataSummaries = new
ArrayList<>();

        responses.stream().forEach(response -> imageSetsMetadataSummaries
            .addAll(response.imageSetsMetadataSummaries()));

        return imageSetsMetadataSummaries;
    } catch (MedicalImagingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }

    return null;
}
```

Caso d'uso #1: operatore EQUAL.



```

List<SearchFilter> searchFilters =
Collections.singletonList(SearchFilter.builder()
    .operator(Operator.EQUAL)
    .values(SearchByAttributeValue.builder()
        .dicomPatientId(patientId)
        .build())
    .build());

SearchCriteria searchCriteria = SearchCriteria.builder()
    .filters(searchFilters)
    .build();

List<ImageSetsMetadataSummary> imageSetsMetadataSummaries =
searchMedicalImagingImageSets(
    medicalImagingClient,
    datastoreId, searchCriteria);
if (imageSetsMetadataSummaries != null) {
    System.out.println("The image sets for patient " + patientId + " are:\n"
        + imageSetsMetadataSummaries);
    System.out.println();
}

```

Caso d'uso #2: operatore BETWEEN che utilizza DICOM StudyDate e DICOMStudyTime.

```

DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyyMMdd");
searchFilters = Collections.singletonList(SearchFilter.builder()
    .operator(Operator.BETWEEN)
    .values(SearchByAttributeValue.builder()

.dicomStudyDateAndTime(DICOMStudyDateAndTime.builder()
    .dicomStudyDate("19990101")
    .dicomStudyTime("000000.000")
    .build())
    .build(),
    SearchByAttributeValue.builder()

.dicomStudyDateAndTime(DICOMStudyDateAndTime.builder()
    .dicomStudyDate((LocalDate.now()
        .format(formatter)))
    .dicomStudyTime("000000.000")
    .build())
    .build())

```

```

        .build());

searchCriteria = SearchCriteria.builder()
    .filters(searchFilters)
    .build();

imageSetsMetadataSummaries =
searchMedicalImagingImageSets(medicalImagingClient,
    datastoreId, searchCriteria);
if (imageSetsMetadataSummaries != null) {
    System.out.println(
        "The image sets searched with BETWEEN operator using
DICOMStudyDate and DICOMStudyTime are:\n"
        +
        imageSetsMetadataSummaries);
    System.out.println();
}

```

Caso d'uso #3: operatore BETWEEN che utilizza CreateDat. Gli studi sul tempo erano stati precedentemente proseguiti.

```

searchFilters = Collections.singletonList(SearchFilter.builder()
    .operator(Operator.BETWEEN)
    .values(SearchByAttributeValue.builder()
        .createdAt(Instant.parse("1985-04-12T23:20:50.52Z"))
        .build(),
        SearchByAttributeValue.builder()
        .createdAt(Instant.now())
        .build())
    .build());

searchCriteria = SearchCriteria.builder()
    .filters(searchFilters)
    .build();
imageSetsMetadataSummaries =
searchMedicalImagingImageSets(medicalImagingClient,
    datastoreId, searchCriteria);
if (imageSetsMetadataSummaries != null) {
    System.out.println("The image sets searched with BETWEEN operator using
createdAt are:\n "
        + imageSetsMetadataSummaries);
    System.out.println();
}

```

```
}

```

Caso d'uso #4: operatore EQUAL su DICOM SeriesInstance UID e BETWEEN su updatedAt e ordina la risposta in ordine ASC nel campo updatedAt.

```
Instant startDate = Instant.parse("1985-04-12T23:20:50.52Z");
Instant endDate = Instant.now();

searchFilters = Arrays.asList(
    SearchFilter.builder()
        .operator(Operator.EQUAL)
        .values(SearchByAttributeValue.builder()
            .dicomSeriesInstanceUID(seriesInstanceUID)
            .build())
        .build(),
    SearchFilter.builder()
        .operator(Operator.BETWEEN)
        .values(
            SearchByAttributeValue.builder().updatedAt(startDate).build(),
            SearchByAttributeValue.builder().updatedAt(endDate).build()
        ).build());

Sort sort =
Sort.builder().sortOrder(SortOrder.ASC).sortField(SortField.UPDATED_AT).build();

searchCriteria = SearchCriteria.builder()
    .filters(searchFilters)
    .sort(sort)
    .build();

imageSetsMetadataSummaries =
searchMedicalImagingImageSets(medicalImagingClient,
    datastoreId, searchCriteria);
if (imageSetsMetadataSummaries != null) {
    System.out.println("The image sets searched with EQUAL operator on
DICOMSeriesInstanceUID and BETWEEN on updatedAt and sort response\n" +
        "in ASC order on updatedAt field are:\n "
        + imageSetsMetadataSummaries);
    System.out.println();
}

```

- Per i dettagli sull'API, vedere in API Reference. [SearchImageSets](#) AWS SDK for Java 2.x

### Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

## StartDICOMImportJob

Il seguente esempio di codice mostra come usare `StartDICOMImportJob`.

SDK per Java 2.x

```
public static String startDicomImportJob(MedicalImagingClient
medicalImagingClient,
    String jobName,
    String datastoreId,
    String dataAccessRoleArn,
    String inputS3Uri,
    String outputS3Uri) {

    try {
        StartDicomImportJobRequest startDicomImportJobRequest =
StartDicomImportJobRequest.builder()
            .jobName(jobName)
            .datastoreId(datastoreId)
            .dataAccessRoleArn(dataAccessRoleArn)
            .inputS3Uri(inputS3Uri)
            .outputS3Uri(outputS3Uri)
            .build();

        StartDicomImportJobResponse response =
medicalImagingClient.startDICOMImportJob(startDicomImportJobRequest);
        return response.jobId();
    } catch (MedicalImagingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }

    return "";
}
```

```
}
```

- Per i dettagli sull'API, consulta [StartDICOM ImportJob](#) in AWS SDK for Java 2.x API Reference.

### Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

## TagResource

Il seguente esempio di codice mostra come usare `TagResource`.

SDK per Java 2.x

```
public static void tagMedicalImagingResource(MedicalImagingClient
medicalImagingClient,
    String resourceArn,
    Map<String, String> tags) {
    try {
        TagResourceRequest tagResourceRequest = TagResourceRequest.builder()
            .resourceArn(resourceArn)
            .tags(tags)
            .build();

        medicalImagingClient.tagResource(tagResourceRequest);

        System.out.println("Tags have been added to the resource.");
    } catch (MedicalImagingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Per i dettagli sull'API, consulta la [TagResource](#) sezione AWS SDK for Java 2.x API Reference.

**Note**

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

## UntagResource

Il seguente esempio di codice mostra come usare `UntagResource`.

SDK per Java 2.x

```
public static void untagMedicalImagingResource(MedicalImagingClient
medicalImagingClient,
        String resourceArn,
        Collection<String> tagKeys) {
    try {
        UntagResourceRequest untagResourceRequest =
UntagResourceRequest.builder()
            .resourceArn(resourceArn)
            .tagKeys(tagKeys)
            .build();

        medicalImagingClient.untagResource(untagResourceRequest);

        System.out.println("Tags have been removed from the resource.");
    } catch (MedicalImagingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Per i dettagli sull'API, consulta la [UntagResource](#) sezione AWS SDK for Java 2.x API Reference.

**Note**

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

## UpdateImageSetMetadata

Il seguente esempio di codice mostra come usare `UpdateImageSetMetadata`.

SDK per Java 2.x

```

    public static void updateMedicalImageSetMetadata(MedicalImagingClient
medicalImagingClient,
                                                    String datastoreId,
                                                    String imagesetId,
                                                    String versionId,
                                                    MetadataUpdates
metadataUpdates) {
    try {
        UpdateImageSetMetadataRequest updateImageSetMetadataRequest =
UpdateImageSetMetadataRequest
        .builder()
        .datastoreId(datastoreId)
        .imageSetId(imagesetId)
        .latestVersionId(versionId)
        .updateImageSetMetadataUpdates(metadataUpdates)
        .build();

        UpdateImageSetMetadataResponse response =
medicalImagingClient.updateImageSetMetadata(updateImageSetMetadataRequest);

        System.out.println("The image set metadata was updated" + response);
    } catch (MedicalImagingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

```

Caso d'uso #1: Inserimento o aggiornamento di un attributo.

```

final String insertAttributes = ""
    {
        "SchemaVersion": 1.1,
        "Study": {
            "DICOM": {
                "StudyDescription": "CT CHEST"
            }
        }
    }

```

```

        }
    }
    """;

    MetadataUpdates metadataInsertUpdates = MetadataUpdates.builder()
        .dicomUpdates(DICOMUpdates.builder()
            .updateableAttributes(SdkBytes.fromByteBuffer(
                ByteBuffer.wrap(insertAttributes
                    .getBytes(StandardCharsets.UTF_8))))
            .build())
        .build();

    updateMedicalImageSetMetadata(medicalImagingClient, datastoreId,
        imagesetId,
        versionid, metadataInsertUpdates);

```

### Caso d'uso #2: rimuovere un attributo.

```

    final String removeAttributes = ""
        {
            "SchemaVersion": 1.1,
            "Study": {
                "DICOM": {
                    "StudyDescription": "CT CHEST"
                }
            }
        }
    """;

    MetadataUpdates metadataRemoveUpdates = MetadataUpdates.builder()
        .dicomUpdates(DICOMUpdates.builder()
            .removableAttributes(SdkBytes.fromByteBuffer(
                ByteBuffer.wrap(removeAttributes
                    .getBytes(StandardCharsets.UTF_8))))
            .build())
        .build();

    updateMedicalImageSetMetadata(medicalImagingClient, datastoreId,
        imagesetId,
        versionid, metadataRemoveUpdates);

```

### Caso d'uso #3: rimuove un'istanza.



```

final String removeInstance = ""
    {
        "SchemaVersion": 1.1,
        "Study": {
            "Series": {
                "1.1.1.1.1.1.12345.123456789012.123.12345678901234.1": {
                    "Instances": {
                        "1.1.1.1.1.1.12345.123456789012.123.12345678901234.1":
                    }
                }
            }
        }
    }
    """;
MetadataUpdates metadataRemoveUpdates = MetadataUpdates.builder()
    .dicomUpdates(DICOMUpdates.builder()
        .removableAttributes(SdkBytes.fromByteBuffer(
            ByteBuffer.wrap(removeInstance
                .getBytes(StandardCharsets.UTF_8))))
        .build())
    .build();

updateMedicalImageSetMetadata(medicalImagingClient, datastoreId,
    imagesetId,
        versionid, metadataRemoveUpdates);

```

- Per i dettagli sull'API, consulta la [UpdateImageSetMetadata](#) sezione AWS SDK for Java 2.x API Reference.

### Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

## Scenari

### Taggare un archivio dati

Il seguente esempio di codice mostra come etichettare un HealthImaging data store.

## SDK per Java 2.x

Per etichettare un archivio dati.

```
final String datastoreArn = "arn:aws:medical-imaging:us-  
east-1:123456789012:datastore/12345678901234567890123456789012";  
  
TagResource.tagMedicalImagingResource(medicalImagingClient,  
datastoreArn,  
ImmutableMap.of("Deployment", "Development"));
```

La funzione di utilità per etichettare una risorsa.

```
public static void tagMedicalImagingResource(MedicalImagingClient  
medicalImagingClient,  
String resourceArn,  
Map<String, String> tags) {  
    try {  
        TagResourceRequest tagResourceRequest = TagResourceRequest.builder()  
            .resourceArn(resourceArn)  
            .tags(tags)  
            .build();  
  
        medicalImagingClient.tagResource(tagResourceRequest);  
  
        System.out.println("Tags have been added to the resource.");  
    } catch (MedicalImagingException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

Per elencare i tag per un archivio dati.

```
final String datastoreArn = "arn:aws:medical-imaging:us-  
east-1:123456789012:datastore/12345678901234567890123456789012";  
  
ListTagsForResourceResponse result =  
ListTagsForResource.listMedicalImagingResourceTags(  
    medicalImagingClient,  
    datastoreArn);
```

```
        if (result != null) {
            System.out.println("Tags for resource: " + result.tags());
        }
    }
```

La funzione di utilità per elencare i tag di una risorsa.

```
public static ListTagsForResourceResponse
listMedicalImagingResourceTags(MedicalImagingClient medicalImagingClient,
    String resourceArn) {
    try {
        ListTagsForResourceRequest listTagsForResourceRequest =
ListTagsForResourceRequest.builder()
            .resourceArn(resourceArn)
            .build();

        return
medicalImagingClient.listTagsForResource(listTagsForResourceRequest);
    } catch (MedicalImagingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }

    return null;
}
```

Per rimuovere i tag da un archivio dati.

```
final String datastoreArn = "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012";

UntagResource.untagMedicalImagingResource(medicalImagingClient,
    datastoreArn,
        Collections.singletonList("Deployment"));
```

La funzione di utilità per rimuovere il tag di una risorsa.

```
public static void untagMedicalImagingResource(MedicalImagingClient
medicalImagingClient,
    String resourceArn,
```

```
        Collection<String> tagKeys) {
    try {
        UntagResourceRequest untagResourceRequest =
UntagResourceRequest.builder()
            .resourceArn(resourceArn)
            .tagKeys(tagKeys)
            .build();

        medicalImagingClient.untagResource(untagResourceRequest);

        System.out.println("Tags have been removed from the resource.");
    } catch (MedicalImagingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella Documentazione di riferimento delle API AWS SDK for Java 2.x .
  - [ListTagsForResource](#)
  - [TagResource](#)
  - [UntagResource](#)

#### Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

## Taggare un set di immagini

Il seguente esempio di codice mostra come etichettare un set di HealthImaging immagini.

### SDK per Java 2.x

Per etichettare un set di immagini.

```
        final String imageSetArn = "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012/
imageset/12345678901234567890123456789012";
```

```
TagResource.tagMedicalImagingResource(medicalImagingClient,  
imageSetArn,  
ImmutableMap.of("Deployment", "Development"));
```

La funzione di utilità per etichettare una risorsa.

```
public static void tagMedicalImagingResource(MedicalImagingClient  
medicalImagingClient,  
String resourceArn,  
Map<String, String> tags) {  
    try {  
        TagResourceRequest tagResourceRequest = TagResourceRequest.builder()  
            .resourceArn(resourceArn)  
            .tags(tags)  
            .build();  
  
        medicalImagingClient.tagResource(tagResourceRequest);  
  
        System.out.println("Tags have been added to the resource.");  
    } catch (MedicalImagingException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

Per elencare i tag per un set di immagini.

```
final String imageSetArn = "arn:aws:medical-imaging:us-  
east-1:123456789012: datastore/12345678901234567890123456789012/  
imageset/12345678901234567890123456789012";  
  
ListTagsForResourceResponse result =  
ListTagsForResource.listMedicalImagingResourceTags(  
    medicalImagingClient,  
    imageSetArn);  
if (result != null) {  
    System.out.println("Tags for resource: " + result.tags());  
}
```

La funzione di utilità per elencare i tag di una risorsa.

```
public static ListTagsForResourceResponse
listMedicalImagingResourceTags(MedicalImagingClient medicalImagingClient,
    String resourceArn) {
    try {
        ListTagsForResourceRequest listTagsForResourceRequest =
ListTagsForResourceRequest.builder()
            .resourceArn(resourceArn)
            .build();

        return
medicalImagingClient.listTagsForResource(listTagsForResourceRequest);
    } catch (MedicalImagingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }

    return null;
}
```

Per rimuovere i tag da un set di immagini.

```
final String imageSetArn = "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012/
imageset/12345678901234567890123456789012";

UntagResource.untagMedicalImagingResource(medicalImagingClient,
    imageSetArn,
        Collections.singletonList("Deployment"));
```

La funzione di utilità per rimuovere il tag di una risorsa.

```
public static void untagMedicalImagingResource(MedicalImagingClient
medicalImagingClient,
    String resourceArn,
    Collection<String> tagKeys) {
    try {
        UntagResourceRequest untagResourceRequest =
UntagResourceRequest.builder()
```

```
        .resourceArn(resourceArn)
        .tagKeys(tagKeys)
        .build();

    medicalImagingClient.untagResource(untagResourceRequest);

    System.out.println("Tags have been removed from the resource.");
} catch (MedicalImagingException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella Documentazione di riferimento delle API AWS SDK for Java 2.x .
  - [ListTagsForResource](#)
  - [TagResource](#)
  - [UntagResource](#)

#### Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

## Esempi IAM che utilizzano SDK for Java 2.x

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando AWS SDK for Java 2.x with IAM.

Le operazioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le operazioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati e negli esempi tra servizi.

Scenari: esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio.

Ogni esempio include un collegamento a GitHub, dove è possibile trovare istruzioni su come configurare ed eseguire il codice nel contesto.

## Nozioni di base

### Hello IAM

Gli esempi di codice seguenti mostrano come iniziare a utilizzare IAM.

### SDK per Java 2.x

#### Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.ListPoliciesResponse;
import software.amazon.awssdk.services.iam.model.Policy;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class HelloIAM {
    public static void main(String[] args) {
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
            .region(region)
            .build();

        listPolicies(iam);
    }

    public static void listPolicies(IamClient iam) {
        ListPoliciesResponse response = iam.listPolicies();
        List<Policy> polList = response.policies();
        polList.forEach(policy -> {
            System.out.println("Policy Name: " + policy.policyName());
        });
    }
}
```



```
    });  
  }  
}
```

- Per i dettagli sull'API, [ListPolicies](#) consulta AWS SDK for Java 2.x API Reference.

## Argomenti

- [Azioni](#)
- [Scenari](#)

## Azioni

### **AttachRolePolicy**

Il seguente esempio di codice mostra come utilizzare `AttachRolePolicy`.

#### SDK per Java 2.x

#### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;  
import software.amazon.awssdk.services.iam.IamClient;  
import software.amazon.awssdk.services.iam.model.IamException;  
import software.amazon.awssdk.services.iam.model.AttachRolePolicyRequest;  
import software.amazon.awssdk.services.iam.model.AttachedPolicy;  
import software.amazon.awssdk.services.iam.model.ListAttachedRolePoliciesRequest;  
import software.amazon.awssdk.services.iam.model.ListAttachedRolePoliciesResponse;  
import java.util.List;  
  
/**  
 * Before running this Java V2 code example, set up your development  
 * environment, including your credentials.  
 *  
 * For more information, see the following documentation topic:  
 */
```

```
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class AttachRolePolicy {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <roleName> <policyArn>\s

                Where:
                roleName - A role name that you can obtain from the AWS
Management Console.\s
                policyArn - A policy ARN that you can obtain from the AWS
Management Console.\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String roleName = args[0];
        String policyArn = args[1];

        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
            .region(region)
            .build();

        attachIAMRolePolicy(iam, roleName, policyArn);
        iam.close();
    }

    public static void attachIAMRolePolicy(IamClient iam, String roleName, String
policyArn) {
        try {
            ListAttachedRolePoliciesRequest request =
ListAttachedRolePoliciesRequest.builder()
                .roleName(roleName)
                .build();

            ListAttachedRolePoliciesResponse response =
iam.listAttachedRolePolicies(request);
            List<AttachedPolicy> attachedPolicies = response.attachedPolicies();

```

```
        // Ensure that the policy is not attached to this role
        String polArn = "";
        for (AttachedPolicy policy : attachedPolicies) {
            polArn = policy.policyArn();
            if (polArn.compareTo(policyArn) == 0) {
                System.out.println(roleName + " policy is already attached to
this role.");
                return;
            }
        }

        AttachRolePolicyRequest attachRequest =
AttachRolePolicyRequest.builder()
            .roleName(roleName)
            .policyArn(policyArn)
            .build();

        iam.attachRolePolicy(attachRequest);

        System.out.println("Successfully attached policy " + policyArn +
            " to role " + roleName);

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    System.out.println("Done");
}
}
```

- Per i dettagli sull'API, [AttachRolePolicy](#) consulta AWS SDK for Java 2.x API Reference.

## CreateAccessKey

Il seguente esempio di codice mostra come utilizzare `CreateAccessKey`.

## SDK per Java 2.x

 Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.services.iam.model.CreateAccessKeyRequest;
import software.amazon.awssdk.services.iam.model.CreateAccessKeyResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateAccessKey {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <user>\s

                Where:
                user - An AWS IAM user that you can obtain from the AWS
Management Console.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String user = args[0];
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
```

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

    String keyId = createIAMAccessKey(iam, user);
    System.out.println("The Key Id is " + keyId);
    iam.close();
}

public static String createIAMAccessKey(IamClient iam, String user) {
    try {
        CreateAccessKeyRequest request = CreateAccessKeyRequest.builder()
            .userName(user)
            .build();

        CreateAccessKeyResponse response = iam.createAccessKey(request);
        return response.accessKey().accessKeyId();

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
}
```

- Per i dettagli sull'API, [CreateAccessKey](#) consulta AWS SDK for Java 2.x API Reference.

## CreateAccountAlias

Il seguente esempio di codice mostra come utilizzare `CreateAccountAlias`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.services.iam.model.CreateAccountAliasRequest;
import software.amazon.awssdk.regions.Region;
```

```
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateAccountAlias {
    public static void main(String[] args) {
        final String usage = ""
            Usage:
                <alias>\s

            Where:
                alias - The account alias to create (for example, myawsaccount).
\s

            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String alias = args[0];
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
            .region(region)
            .build();

        createIAMAccountAlias(iam, alias);
        iam.close();
        System.out.println("Done");
    }

    public static void createIAMAccountAlias(IamClient iam, String alias) {
        try {
            CreateAccountAliasRequest request = CreateAccountAliasRequest.builder()
                .accountAlias(alias)
                .build();
        }
    }
}
```

```
        iam.createAccountAlias(request);
        System.out.println("Successfully created account alias: " + alias);

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Per i dettagli sull'API, [CreateAccountAlias](#) consulta AWS SDK for Java 2.x API Reference.

## CreatePolicy

Il seguente esempio di codice mostra come utilizzare `CreatePolicy`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.services.iam.model.CreatePolicyRequest;
import software.amazon.awssdk.services.iam.model.CreatePolicyResponse;
import software.amazon.awssdk.services.iam.model.GetPolicyRequest;
import software.amazon.awssdk.services.iam.model.GetPolicyResponse;
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.waiters.IamWaiter;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
```

```

*/
public class CreatePolicy {

    public static final String PolicyDocument = "{" +
        "  \"Version\": \"2012-10-17\", " +
        "  \"Statement\": [ " +
        "    { " +
        "      \"Effect\": \"Allow\", " +
        "      \"Action\": [ " +
        "        \"dynamodb:DeleteItem\", " +
        "        \"dynamodb:GetItem\", " +
        "        \"dynamodb:PutItem\", " +
        "        \"dynamodb:Scan\", " +
        "        \"dynamodb:UpdateItem\" " +
        "      ], " +
        "      \"Resource\": \"*\"/>
    "    ] " +
    "  }";

    public static void main(String[] args) {

        final String usage = ""
            Usage:
              CreatePolicy <policyName>\s

            Where:
              policyName - A unique policy name.\s
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String policyName = args[0];
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
            .region(region)
            .build();

        String result = createIAMPolicy(iam, policyName);
        System.out.println("Successfully created a policy with this ARN value: " +
            result);
    }
}

```



```
        iam.close();
    }

    public static String createIAMPolicy(IamClient iam, String policyName) {
        try {
            // Create an IamWaiter object.
            IamWaiter iamWaiter = iam.waiter();

            CreatePolicyRequest request = CreatePolicyRequest.builder()
                .policyName(policyName)
                .policyDocument(PolicyDocument)
                .build();

            CreatePolicyResponse response = iam.createPolicy(request);

            // Wait until the policy is created.
            GetPolicyRequest polRequest = GetPolicyRequest.builder()
                .policyArn(response.policy().arn())
                .build();

            WaiterResponse<GetPolicyResponse> waitUntilPolicyExists =
iamWaiter.waitUntilPolicyExists(polRequest);

            waitUntilPolicyExists.matched().response().ifPresent(System.out::println);
            return response.policy().arn();

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
}
```

- Per i dettagli sull'API, [CreatePolicy](#) consulta AWS SDK for Java 2.x API Reference.

## CreateRole

Il seguente esempio di codice mostra come utilizzare `CreateRole`.

## SDK per Java 2.x

 Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import org.json.simple.JSONObject;
import org.json.simple.parser.JSONParser;
import software.amazon.awssdk.services.iam.model.CreateRoleRequest;
import software.amazon.awssdk.services.iam.model.CreateRoleResponse;
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import java.io.FileReader;

/*
 * This example requires a trust policy document. For more information, see:
 * https://aws.amazon.com/blogs/security/how-to-use-trust-policies-with-iam-roles/
 *
 * In addition, set up your development environment, including your credentials.
 *
 * For information, see this documentation topic:
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class CreateRole {
    public static void main(String[] args) throws Exception {
        final String usage = ""
            Usage:
                <rolename> <fileLocation>\s

                Where:
                    rolename - The name of the role to create.\s
                    fileLocation - The location of the JSON document that represents
the trust policy.\s
            """;

        if (args.length != 2) {
```

```

        System.out.println(usage);
        System.exit(1);
    }

    String rolename = args[0];
    String fileLocation = args[1];
    Region region = Region.AWS_GLOBAL;
    IamClient iam = IamClient.builder()
        .region(region)
        .build();

    String result = createIAMRole(iam, rolename, fileLocation);
    System.out.println("Successfully created user: " + result);
    iam.close();
}

public static String createIAMRole(IamClient iam, String rolename, String
fileLocation) throws Exception {
    try {
        JSONObject jsonObject = (JSONObject) readJsonSimpleDemo(fileLocation);
        CreateRoleRequest request = CreateRoleRequest.builder()
            .roleName(rolename)
            .assumeRolePolicyDocument(jsonObject.toJSONString())
            .description("Created using the AWS SDK for Java")
            .build();

        CreateRoleResponse response = iam.createRole(request);
        System.out.println("The ARN of the role is " + response.role().arn());

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}

public static Object readJsonSimpleDemo(String filename) throws Exception {
    FileReader reader = new FileReader(filename);
    JSONParser jsonParser = new JSONParser();
    return jsonParser.parse(reader);
}
}

```

- Per i dettagli sull'API, [CreateRole](#) consulta AWS SDK for Java 2.x API Reference.

## CreateUser

Il seguente esempio di codice mostra come utilizzare CreateUser.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.services.iam.model.CreateUserRequest;
import software.amazon.awssdk.services.iam.model.CreateUserResponse;
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.waiters.IamWaiter;
import software.amazon.awssdk.services.iam.model.GetUserRequest;
import software.amazon.awssdk.services.iam.model.GetUserResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateUser {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <username>\s

                Where:
                username - The name of the user to create.\s

                """;
```

```
    if (args.length != 1) {
        System.out.println(usage);
        System.exit(1);
    }

    String username = args[0];
    Region region = Region.AWS_GLOBAL;
    IamClient iam = IamClient.builder()
        .region(region)
        .build();

    String result = createIAMUser(iam, username);
    System.out.println("Successfully created user: " + result);
    iam.close();
}

public static String createIAMUser(IamClient iam, String username) {
    try {
        // Create an IamWaiter object.
        IamWaiter iamWaiter = iam.waiter();

        CreateUserRequest request = CreateUserRequest.builder()
            .userName(username)
            .build();

        CreateUserResponse response = iam.createUser(request);

        // Wait until the user is created.
        GetUserRequest userRequest = GetUserRequest.builder()
            .userName(response.user().userName())
            .build();

        WaiterResponse<GetUserResponse> waitUntilUserExists =
iamWaiter.waitUntilUserExists(userRequest);
        waitUntilUserExists.matched().response().ifPresent(System.out::println);
        return response.user().userName();

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
```

```
}
```

- Per i dettagli sull'API, [CreateUser](#) consulta AWS SDK for Java 2.x API Reference.

## DeleteAccessKey

Il seguente esempio di codice mostra come utilizzare `DeleteAccessKey`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.DeleteAccessKeyRequest;
import software.amazon.awssdk.services.iam.model.IamException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteAccessKey {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <username> <accessKey>\s

                Where:
                username - The name of the user.\s
                accessKey - The access key ID for the secret access key you want
                to delete.\s
    }
}
```

```
        """);

    if (args.length != 2) {
        System.out.println(usage);
        System.exit(1);
    }

    String username = args[0];
    String accessKey = args[1];
    Region region = Region.AWS_GLOBAL;
    IamClient iam = IamClient.builder()
        .region(region)
        .build();
    deleteKey(iam, username, accessKey);
    iam.close();
}

public static void deleteKey(IamClient iam, String username, String accessKey) {
    try {
        DeleteAccessKeyRequest request = DeleteAccessKeyRequest.builder()
            .accessKeyId(accessKey)
            .userName(username)
            .build();

        iam.deleteAccessKey(request);
        System.out.println("Successfully deleted access key " + accessKey +
            " from user " + username);

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Per i dettagli sull'API, [DeleteAccessKey](#) consulta AWS SDK for Java 2.x API Reference.

## DeleteAccountAlias

Il seguente esempio di codice mostra come utilizzare `DeleteAccountAlias`.

## SDK per Java 2.x

 Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.services.iam.model.DeleteAccountAliasRequest;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteAccountAlias {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <alias>\s

                Where:
                alias - The account alias to delete.\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String alias = args[0];
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
            .region(region)
            .build();
```



```
        deleteIAMAccountAlias(iam, alias);
        iam.close();
    }

    public static void deleteIAMAccountAlias(IamClient iam, String alias) {
        try {
            DeleteAccountAliasRequest request = DeleteAccountAliasRequest.builder()
                .accountAlias(alias)
                .build();

            iam.deleteAccountAlias(request);
            System.out.println("Successfully deleted account alias " + alias);

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        System.out.println("Done");
    }
}
```

- Per i dettagli sull'API, [DeleteAccountAlias](#) consulta AWS SDK for Java 2.x API Reference.

## DeletePolicy

Il seguente esempio di codice mostra come utilizzare DeletePolicy.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.services.iam.model.DeletePolicyRequest;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;
```

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeletePolicy {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <policyARN>\s

            Where:
                policyARN - A policy ARN value to delete.\s
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String policyARN = args[0];
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
            .region(region)
            .build();

        deleteIAMPolicy(iam, policyARN);
        iam.close();
    }

    public static void deleteIAMPolicy(IamClient iam, String policyARN) {
        try {
            DeletePolicyRequest request = DeletePolicyRequest.builder()
                .policyArn(policyARN)
                .build();

            iam.deletePolicy(request);
            System.out.println("Successfully deleted the policy");
        } catch (IamException e) {
```

```
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    System.out.println("Done");
}
}
```

- Per i dettagli sull'API, [DeletePolicy](#) consulta AWS SDK for Java 2.x API Reference.

## DeleteUser

Il seguente esempio di codice mostra come utilizzare `DeleteUser`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.DeleteUserRequest;
import software.amazon.awssdk.services.iam.model.IamException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteUser {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <userName>\s
```

```
        Where:
            userName - The name of the user to delete.\s
        """;

    if (args.length != 1) {
        System.out.println(usage);
        System.exit(1);
    }

    String userName = args[0];
    Region region = Region.AWS_GLOBAL;
    IamClient iam = IamClient.builder()
        .region(region)
        .build();

    deleteIAMUser(iam, userName);
    System.out.println("Done");
    iam.close();
}

public static void deleteIAMUser(IamClient iam, String userName) {
    try {
        DeleteUserRequest request = DeleteUserRequest.builder()
            .userName(userName)
            .build();

        iam.deleteUser(request);
        System.out.println("Successfully deleted IAM user " + userName);

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Per i dettagli sull'API, [DeleteUser](#) consulta AWS SDK for Java 2.x API Reference.

## DetachRolePolicy

Il seguente esempio di codice mostra come utilizzare `DetachRolePolicy`.

## SDK per Java 2.x

 Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.services.iam.model.DetachRolePolicyRequest;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DetachRolePolicy {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <roleName> <policyArn>\s

            Where:
                roleName - A role name that you can obtain from the AWS
Management Console.\s
                policyArn - A policy ARN that you can obtain from the AWS
Management Console.\s
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String roleName = args[0];
        String policyArn = args[1];
```

```
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
            .region(region)
            .build();
        detachPolicy(iam, roleName, policyArn);
        System.out.println("Done");
        iam.close();
    }

    public static void detachPolicy(IamClient iam, String roleName, String
policyArn) {
        try {
            DetachRolePolicyRequest request = DetachRolePolicyRequest.builder()
                .roleName(roleName)
                .policyArn(policyArn)
                .build();

            iam.detachRolePolicy(request);
            System.out.println("Successfully detached policy " + policyArn +
                " from role " + roleName);

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Per i dettagli sull'API, [DetachRolePolicy](#) consulta AWS SDK for Java 2.x API Reference.

## ListAccessKeys

Il seguente esempio di codice mostra come utilizzare `ListAccessKeys`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.services.iam.model.AccessKeyMetadata;
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.services.iam.model.ListAccessKeysRequest;
import software.amazon.awssdk.services.iam.model.ListAccessKeysResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListAccessKeys {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <userName>\s

                Where:
                userName - The name of the user for which access keys are
retrieved.\s

                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String userName = args[0];
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
                .region(region)
                .build();

        listKeys(iam, userName);
        System.out.println("Done");
        iam.close();
    }
}
```

```
public static void listKeys(IamClient iam, String userName) {
    try {
        boolean done = false;
        String newMarker = null;

        while (!done) {
            ListAccessKeysResponse response;

            if (newMarker == null) {
                ListAccessKeysRequest request = ListAccessKeysRequest.builder()
                    .userName(userName)
                    .build();

                response = iam.listAccessKeys(request);
            } else {
                ListAccessKeysRequest request = ListAccessKeysRequest.builder()
                    .userName(userName)
                    .marker(newMarker)
                    .build();

                response = iam.listAccessKeys(request);
            }

            for (AccessKeyMetadata metadata : response.accessKeyMetadata()) {
                System.out.format("Retrieved access key %s",
metadata.accessKeyId());
            }

            if (!response.isTruncated()) {
                done = true;
            } else {
                newMarker = response.marker();
            }
        }

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```



- Per i dettagli sull'API, [ListAccessKeys](#) consulta AWS SDK for Java 2.x API Reference.

## ListAccountAliases

Il seguente esempio di codice mostra come utilizzare `ListAccountAliases`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.services.iam.model.ListAccountAliasesResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListAccountAliases {
    public static void main(String[] args) {
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
            .region(region)
            .build();

        listAliases(iam);
        System.out.println("Done");
        iam.close();
    }

    public static void listAliases(IamClient iam) {
        try {
            ListAccountAliasesResponse response = iam.listAccountAliases();
        }
    }
}
```

```

        for (String alias : response.accountAliases()) {
            System.out.printf("Retrieved account alias %s", alias);
        }

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
}

```

- Per i dettagli sull'API, [ListAccountAliases](#) consulta AWS SDK for Java 2.x API Reference.

## ListUsers

Il seguente esempio di codice mostra come utilizzare `ListUsers`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```

import software.amazon.awssdk.services.iam.model.AttachedPermissionsBoundary;
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.services.iam.model.ListUsersRequest;
import software.amazon.awssdk.services.iam.model.ListUsersResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.User;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

```

```
public class ListUsers {
    public static void main(String[] args) {
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
            .region(region)
            .build();

        listAllUsers(iam);
        System.out.println("Done");
        iam.close();
    }

    public static void listAllUsers(IamClient iam) {
        try {
            boolean done = false;
            String newMarker = null;
            while (!done) {
                ListUsersResponse response;
                if (newMarker == null) {
                    ListUsersRequest request = ListUsersRequest.builder().build();
                    response = iam.listUsers(request);
                } else {
                    ListUsersRequest request = ListUsersRequest.builder()
                        .marker(newMarker)
                        .build();

                    response = iam.listUsers(request);
                }

                for (User user : response.users()) {
                    System.out.format("\n Retrieved user %s", user.userName());
                    AttachedPermissionsBoundary permissionsBoundary =
user.permissionsBoundary();
                    if (permissionsBoundary != null)
                        System.out.format("\n Permissions boundary details %s",
permissionsBoundary.permissionsBoundaryTypeAsString());
                }

                if (!response.isTruncated()) {
                    done = true;
                } else {
                    newMarker = response.marker();
                }
            }
        }
    }
}
```

```
    }  
  
    } catch (IamException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}  
}
```

- Per i dettagli sull'API, [ListUsers](#) consulta AWS SDK for Java 2.x API Reference.

## UpdateAccessKey

Il seguente esempio di codice mostra come utilizzare `UpdateAccessKey`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.services.iam.model.IamException;  
import software.amazon.awssdk.services.iam.model.StatusType;  
import software.amazon.awssdk.services.iam.model.UpdateAccessKeyRequest;  
import software.amazon.awssdk.regions.Region;  
import software.amazon.awssdk.services.iam.IamClient;  
  
/**  
 * Before running this Java V2 code example, set up your development  
 * environment, including your credentials.  
 *  
 * For more information, see the following documentation topic:  
 *  
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html  
 */  
public class UpdateAccessKey {  
  
    private static StatusType statusType;
```

```

public static void main(String[] args) {
    final String usage = ""

        Usage:
            <username> <accessId> <status>\s

        Where:
            username - The name of the user whose key you want to update.\s
            accessId - The access key ID of the secret access key you want
to update.\s
            status - The status you want to assign to the secret access key.
\s

        """;

    if (args.length != 3) {
        System.out.println(usage);
        System.exit(1);
    }

    String username = args[0];
    String accessId = args[1];
    String status = args[2];
    Region region = Region.AWS_GLOBAL;
    IamClient iam = IamClient.builder()
        .region(region)
        .build();

    updateKey(iam, username, accessId, status);
    System.out.println("Done");
    iam.close();
}

public static void updateKey(IamClient iam, String username, String accessId,
String status) {
    try {
        if (status.toLowerCase().equalsIgnoreCase("active")) {
            statusType = StatusType.ACTIVE;
        } else if (status.toLowerCase().equalsIgnoreCase("inactive")) {
            statusType = StatusType.INACTIVE;
        } else {
            statusType = StatusType.UNKNOWN_TO_SDK_VERSION;
        }

        UpdateAccessKeyRequest request = UpdateAccessKeyRequest.builder()

```

```

        .accessKeyId(accessId)
        .userName(username)
        .status(statusType)
        .build();

    iam.updateAccessKey(request);
    System.out.printf("Successfully updated the status of access key %s to"
+
        "status %s for user %s", accessId, status, username);

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}

```

- Per i dettagli sull'API, [UpdateAccessKey](#) consulta AWS SDK for Java 2.x API Reference.

## UpdateUser

Il seguente esempio di codice mostra come utilizzare `UpdateUser`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.services.iam.model.UpdateUserRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:

```

```
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class UpdateUser {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <curName> <newName>\s

            Where:
                curName - The current user name.\s
                newName - An updated user name.\s
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String curName = args[0];
        String newName = args[1];
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
            .region(region)
            .build();

        updateIAMUser(iam, curName, newName);
        System.out.println("Done");
        iam.close();
    }

    public static void updateIAMUser(IamClient iam, String curName, String newName)
    {
        try {
            UpdateUserRequest request = UpdateUserRequest.builder()
                .userName(curName)
                .newUserName(newName)
                .build();

            iam.updateUser(request);
            System.out.printf("Successfully updated user to username %s", newName);
        } catch (IamException e) {
```

```
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Per i dettagli sull'API, [UpdateUser](#) consulta AWS SDK for Java 2.x API Reference.

## Scenari

### Creazione e gestione di un servizio resiliente

Il seguente esempio di codice mostra come creare un servizio Web con bilanciamento del carico che restituisca consigli su libri, film e canzoni. L'esempio mostra come il servizio risponde ai guasti e spiega come ristrutturarlo per una maggiore resilienza in caso di guasti.

- Utilizza un gruppo con dimensionamento automatico Amazon EC2 per creare istanze Amazon Elastic Compute Cloud (Amazon EC2) basate su un modello di avvio e per mantenere il numero di istanze entro un intervallo specificato.
- Gestisci e distribuisce le richieste HTTP con Elastic Load Balancing.
- Monitora lo stato delle istanze in un gruppo con dimensionamento automatico e inoltra le richieste soltanto alle istanze integre.
- Esegui un server Web Python su ogni istanza EC2 per gestire le richieste HTTP. Il server Web risponde con consigli e controlli dell'integrità.
- Simula un servizio di raccomandazione con una tabella Amazon DynamoDB.
- Controlla la risposta del server web alle richieste e ai controlli sanitari aggiornando AWS Systems Manager i parametri.

### SDK per Java 2.x

#### Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Esegui lo scenario interattivo al prompt dei comandi.



```
public class Main {

    public static final String fileName = "C:\\AWS\\resworkflow\\
\\recommendations.json"; // Modify file location.
    public static final String tableName = "doc-example-recommendation-service";
    public static final String startScript = "C:\\AWS\\resworkflow\\
\\server_startup_script.sh"; // Modify file location.
    public static final String policyFile = "C:\\AWS\\resworkflow\\
\\instance_policy.json"; // Modify file location.
    public static final String ssmJSON = "C:\\AWS\\resworkflow\\
\\ssm_only_policy.json"; // Modify file location.
    public static final String failureResponse = "doc-example-resilient-
architecture-failure-response";
    public static final String healthCheck = "doc-example-resilient-architecture-
health-check";
    public static final String templateName = "doc-example-resilience-template";
    public static final String roleName = "doc-example-resilience-role";
    public static final String policyName = "doc-example-resilience-pol";
    public static final String profileName = "doc-example-resilience-prof";

    public static final String badCredsProfileName = "doc-example-resilience-prof-
bc";

    public static final String targetGroupName = "doc-example-resilience-tg";
    public static final String autoScalingGroupName = "doc-example-resilience-
group";
    public static final String lbName = "doc-example-resilience-lb";
    public static final String protocol = "HTTP";
    public static final int port = 80;

    public static final String DASHES = new String(new char[80]).replace("\\0", "-");

    public static void main(String[] args) throws IOException, InterruptedException
    {
        Scanner in = new Scanner(System.in);
        Database database = new Database();
        AutoScaler autoScaler = new AutoScaler();
        LoadBalancer loadBalancer = new LoadBalancer();

        System.out.println(DASHES);
        System.out.println("Welcome to the demonstration of How to Build and Manage
a Resilient Service!");
        System.out.println(DASHES);
    }
}
```

```
System.out.println(DASHES);
System.out.println("A - SETUP THE RESOURCES");
System.out.println("Press Enter when you're ready to start deploying
resources.");
in.nextLine();
deploy(loadBalancer);
System.out.println(DASHES);
System.out.println(DASHES);
System.out.println("B - DEMO THE RESILIENCE FUNCTIONALITY");
System.out.println("Press Enter when you're ready.");
in.nextLine();
demo(loadBalancer);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("C - DELETE THE RESOURCES");
System.out.println("""
    This concludes the demo of how to build and manage a resilient
service.

    To keep things tidy and to avoid unwanted charges on your account,
we can clean up all AWS resources
    that were created for this demo.
    """);

System.out.println("\n Do you want to delete the resources (y/n)? ");
String userInput = in.nextLine().trim().toLowerCase(); // Capture user input

if (userInput.equals("y")) {
    // Delete resources here
    deleteResources(loadBalancer, autoScaler, database);
    System.out.println("Resources deleted.");
} else {
    System.out.println("""
        Okay, we'll leave the resources intact.
        Don't forget to delete them when you're done with them or you
might incur unexpected charges.
    """);
}
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("The example has completed. ");
System.out.println("\n Thanks for watching!");
```

```

        System.out.println(DASHES);
    }

    // Deletes the AWS resources used in this example.
    private static void deleteResources(LoadBalancer loadBalancer, AutoScaler
autoScaler, Database database)
        throws IOException, InterruptedException {
        loadBalancer.deleteLoadBalancer(lbName);
        System.out.println("*** Wait 30 secs for resource to be deleted");
        TimeUnit.SECONDS.sleep(30);
        loadBalancer.deleteTargetGroup(targetGroupName);
        autoScaler.deleteAutoScaleGroup(autoScalingGroupName);
        autoScaler.deleteRolesPolicies(policyName, roleName, profileName);
        autoScaler.deleteTemplate(templateName);
        database.deleteTable(tableName);
    }

    private static void deploy(LoadBalancer loadBalancer) throws
InterruptedException, IOException {
        Scanner in = new Scanner(System.in);
        System.out.println(
            """
                For this demo, we'll use the AWS SDK for Java (v2) to create
several AWS resources
                to set up a load-balanced web service endpoint and explore
some ways to make it resilient
                against various kinds of failures.

                Some of the resources create by this demo are:
                \t* A DynamoDB table that the web service depends on to
provide book, movie, and song recommendations.
                \t* An EC2 launch template that defines EC2 instances that
each contain a Python web server.
                \t* An EC2 Auto Scaling group that manages EC2 instances
across several Availability Zones.
                \t* An Elastic Load Balancing (ELB) load balancer that
targets the Auto Scaling group to distribute requests.
            """);

        System.out.println("Press Enter when you're ready.");
        in.nextLine();
        System.out.println(DASHES);

        System.out.println(DASHES);

```

```
System.out.println("Creating and populating a DynamoDB table named " +
tableName);
Database database = new Database();
database.createTable(tableName, fileName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("""
    Creating an EC2 launch template that runs '{startup_script}' when an
instance starts.
    This script starts a Python web server defined in the `server.py`
script. The web server
    listens to HTTP requests on port 80 and responds to requests to '/'
and to '/healthcheck'.
    For demo purposes, this server is run as the root user. In
production, the best practice is to
    run a web server, such as Apache, with least-privileged credentials.

    The template also defines an IAM policy that each instance uses to
assume a role that grants
    permissions to access the DynamoDB recommendation table and Systems
Manager parameters
    that control the flow of the demo.
    """);

LaunchTemplateCreator templateCreator = new LaunchTemplateCreator();
templateCreator.createTemplate(policyFile, policyName, profileName,
startScript, templateName, roleName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println(
    "Creating an EC2 Auto Scaling group that maintains three EC2
instances, each in a different Availability Zone.");
System.out.println("*** Wait 30 secs for the VPC to be created");
TimeUnit.SECONDS.sleep(30);
AutoScaler autoScaler = new AutoScaler();
String[] zones = autoScaler.createGroup(3, templateName,
autoScalingGroupName);

System.out.println("""
    At this point, you have EC2 instances created. Once each instance
starts, it listens for
```

```

        HTTP requests. You can see these instances in the console or
continue with the demo.
        Press Enter when you're ready to continue.
        """);

in.nextLine();
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("Creating variables that control the flow of the demo.");
ParameterHelper paramHelper = new ParameterHelper();
paramHelper.reset();
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("""
        Creating an Elastic Load Balancing target group and load balancer.
The target group
        defines how the load balancer connects to instances. The load
balancer provides a
        single endpoint where clients connect and dispatches requests to
instances in the group.
        """);

String vpcId = autoScaler.getDefaultVPC();
List<Subnet> subnets = autoScaler.getSubnets(vpcId, zones);
System.out.println("You have retrieved a list with " + subnets.size() + "
subnets");
String targetGroupArn = loadBalancer.createTargetGroup(protocol, port,
vpcId, targetGroupName);
String elbDnsName = loadBalancer.createLoadBalancer(subnets, targetGroupArn,
lbName, port, protocol);
autoScaler.attachLoadBalancerTargetGroup(autoScalingGroupName,
targetGroupArn);
System.out.println("Verifying access to the load balancer endpoint...");
boolean wasSuccessful = loadBalancer.verifyLoadBalancerEndpoint(elbDnsName);
if (!wasSuccessful) {
    System.out.println("Couldn't connect to the load balancer, verifying
that the port is open...");
    CloseableHttpClient httpClient = HttpClients.createDefault();

    // Create an HTTP GET request to "http://checkip.amazonaws.com"
    HttpGet httpGet = new HttpGet("http://checkip.amazonaws.com");
    try {

```

```

        // Execute the request and get the response
        HttpResponse response = httpClient.execute(httpGet);

        // Read the response content.
        String ipAddress =
IOUtils.toString(response.getEntity().getContent(), StandardCharsets.UTF_8).trim();

        // Print the public IP address.
        System.out.println("Public IP Address: " + ipAddress);
        GroupInfo groupInfo = autoScaler.verifyInboundPort(vpcId, port,
ipAddress);
        if (!groupInfo.isPortOpen()) {
            System.out.println("""
                For this example to work, the default security group for
your default VPC must
                allow access from this computer. You can either add it
automatically from this
                example or add it yourself using the AWS Management
Console.
                """);

            System.out.println(
                "Do you want to add a rule to security group " +
groupInfo.getGroupName() + " to allow");
            System.out.println("inbound traffic on port " + port + " from
your computer's IP address (y/n) ");
            String ans = in.nextLine();
            if ("y".equalsIgnoreCase(ans)) {
                autoScaler.openInboundPort(groupInfo.getGroupName(),
String.valueOf(port), ipAddress);
                System.out.println("Security group rule added.");
            } else {
                System.out.println("No security group rule added.");
            }
        }

    } catch (AutoScalingException e) {
        e.printStackTrace();
    }
    } else if (wasSuccessful) {
        System.out.println("Your load balancer is ready. You can access it by
browsing to:");
        System.out.println("\t http://" + elbDnsName);
    } else {

```

```
        System.out.println("Couldn't get a successful response from the load
balancer endpoint. Troubleshoot by");
        System.out.println("manually verifying that your VPC and security group
are configured correctly and that");
        System.out.println("you can successfully make a GET request to the load
balancer.");
    }

    System.out.println("Press Enter when you're ready to continue with the
demo.");
    in.nextLine();
}

// A method that controls the demo part of the Java program.
public static void demo(LoadBalancer loadBalancer) throws IOException,
InterruptedException {
    ParameterHelper paramHelper = new ParameterHelper();
    System.out.println("Read the ssm_only_policy.json file");
    String ssmOnlyPolicy = readFileAsString(ssmJSON);

    System.out.println("Resetting parameters to starting values for demo.");
    paramHelper.reset();

    System.out.println(
        """
                This part of the demonstration shows how to toggle
different parts of the system
                to create situations where the web service fails, and shows
how using a resilient
                architecture can keep the web service running in spite of
these failures.

                At the start, the load balancer endpoint returns
recommendations and reports that all targets are healthy.
        """);
    demoChoices(loadBalancer);

    System.out.println(
        """
                The web service running on the EC2 instances gets
recommendations by querying a DynamoDB table.
                The table name is contained in a Systems Manager parameter
named self.param_helper.table.
        """);
}
```

```
        To simulate a failure of the recommendation service, let's
set this parameter to name a non-existent table.
        """);
    paramHelper.put(paramHelper.tableName, "this-is-not-a-table");

    System.out.println(
        ""
        \nNow, sending a GET request to the load balancer endpoint
returns a failure code. But, the service reports as
        healthy to the load balancer because shallow health checks
don't check for failure of the recommendation service.
        """);
    demoChoices(loadBalancer);

    System.out.println(
        ""
        Instead of failing when the recommendation service fails,
the web service can return a static response.
        While this is not a perfect solution, it presents the
customer with a somewhat better experience than failure.
        """);
    paramHelper.put(paramHelper.failureResponse, "static");

    System.out.println("""
        Now, sending a GET request to the load balancer endpoint returns a
static response.
        The service still reports as healthy because health checks are still
shallow.
        """);
    demoChoices(loadBalancer);

    System.out.println("Let's reinstate the recommendation service.");
    paramHelper.put(paramHelper.tableName, paramHelper.dyntable);

    System.out.println("""
        Let's also substitute bad credentials for one of the instances in
the target group so that it can't
        access the DynamoDB recommendation table. We will get an instance id
value.
        """);

    LaunchTemplateCreator templateCreator = new LaunchTemplateCreator();
    AutoScaler autoScaler = new AutoScaler();
```



```
// Create a new instance profile based on badCredsProfileName.
templateCreator.createInstanceProfile(policyFile, policyName,
badCredsProfileName, roleName);
String badInstanceId = autoScaler.getBadInstance(autoScalingGroupName);
System.out.println("The bad instance id values used for this demo is " +
badInstanceId);

String profileAssociationId = autoScaler.getInstanceProfile(badInstanceId);
System.out.println("The association Id value is " + profileAssociationId);
System.out.println("Replacing the profile for instance " + badInstanceId
+ " with a profile that contains bad credentials");
autoScaler.replaceInstanceProfile(badInstanceId, badCredsProfileName,
profileAssociationId);

System.out.println(
    """"
        Now, sending a GET request to the load balancer endpoint
returns either a recommendation or a static response,
        depending on which instance is selected by the load
balancer.
    """);

demoChoices(loadBalancer);

System.out.println("""
    Let's implement a deep health check. For this demo, a deep health
check tests whether
    the web service can access the DynamoDB table that it depends on for
recommendations. Note that
    the deep health check is only for ELB routing and not for Auto
Scaling instance health.
    This kind of deep health check is not recommended for Auto Scaling
instance health, because it
    risks accidental termination of all instances in the Auto Scaling
group when a dependent service fails.
    """);

System.out.println("""
    By implementing deep health checks, the load balancer can detect
when one of the instances is failing
    and take that instance out of rotation.
    """);

paramHelper.put(paramHelper.healthCheck, "deep");
```

```
System.out.println("""
    Now, checking target health indicates that the instance with bad
credentials
    is unhealthy. Note that it might take a minute or two for the load
balancer to detect the unhealthy
    instance. Sending a GET request to the load balancer endpoint always
returns a recommendation, because
    the load balancer takes unhealthy instances out of its rotation.
    """);

demoChoices(loadBalancer);

System.out.println(
    """
        Because the instances in this demo are controlled by an auto
scaler, the simplest way to fix an unhealthy
        instance is to terminate it and let the auto scaler start a
new instance to replace it.
        """);
autoScaler.terminateInstance(badInstanceId);

System.out.println("""
    Even while the instance is terminating and the new instance is
starting, sending a GET
    request to the web service continues to get a successful
recommendation response because
    the load balancer routes requests to the healthy instances. After
the replacement instance
    starts and reports as healthy, it is included in the load balancing
rotation.

    Note that terminating and replacing an instance typically takes
several minutes, during which time you
    can see the changing health check status until the new instance is
running and healthy.
    """);

demoChoices(loadBalancer);
System.out.println(
    "If the recommendation service fails now, deep health checks mean
all instances report as unhealthy.");
paramHelper.put(paramHelper.tableName, "this-is-not-a-table");

demoChoices(loadBalancer);
```

```
        paramHelper.reset();
    }

    public static void demoChoices(LoadBalancer loadBalancer) throws IOException,
    InterruptedException {
        String[] actions = {
            "Send a GET request to the load balancer endpoint.",
            "Check the health of load balancer targets.",
            "Go to the next part of the demo."
        };
        Scanner scanner = new Scanner(System.in);

        while (true) {
            System.out.println("-".repeat(88));
            System.out.println("See the current state of the service by selecting
one of the following choices:");
            for (int i = 0; i < actions.length; i++) {
                System.out.println(i + ": " + actions[i]);
            }

            try {
                System.out.print("\nWhich action would you like to take? ");
                int choice = scanner.nextInt();
                System.out.println("-".repeat(88));

                switch (choice) {
                    case 0 -> {
                        System.out.println("Request:\n");
                        System.out.println("GET http://" +
loadBalancer.getEndpoint(lbName));
                        CloseableHttpClient httpClient =
HttpClients.createDefault();

                        // Create an HTTP GET request to the ELB.
                        HttpGet httpGet = new HttpGet("http://" +
loadBalancer.getEndpoint(lbName));

                        // Execute the request and get the response.
                        HttpResponse response = httpClient.execute(httpGet);
                        int statusCode = response.getStatusLine().getStatusCode();
                        System.out.println("HTTP Status Code: " + statusCode);

                        // Display the JSON response
                        BufferedReader reader = new BufferedReader(
```

```

        new
        InputStreamReader(response.getEntity().getContent()));
        StringBuilder jsonResponse = new StringBuilder();
        String line;
        while ((line = reader.readLine()) != null) {
            jsonResponse.append(line);
        }
        reader.close();

        // Print the formatted JSON response.
        System.out.println("Full Response:\n");
        System.out.println(jsonResponse.toString());

        // Close the HTTP client.
        httpClient.close();

    }
    case 1 -> {
        System.out.println("\nChecking the health of load balancer
targets:\n");

        List<TargetHealthDescription> health =
loadBalancer.checkTargetHealth(targetGroupName);
        for (TargetHealthDescription target : health) {
            System.out.printf("\tTarget %s on port %d is %s\n",
target.target().id(),
                                target.target().port(),
target.targetHealth().stateAsString());
        }
        System.out.println("""
check to update
                                Note that it can take a minute or two for the health
                                after changes are made.
                                """);
    }
    case 2 -> {
        System.out.println("\nOkay, let's move on.");
        System.out.println("-".repeat(88));
        return; // Exit the method when choice is 2
    }
    default -> System.out.println("You must choose a value between
0-2. Please select again.");
}

} catch (java.util.InputMismatchException e) {

```

```
        System.out.println("Invalid input. Please select again.");
        scanner.nextLine(); // Clear the input buffer.
    }
}

public static String readFileAsString(String filePath) throws IOException {
    byte[] bytes = Files.readAllBytes(Paths.get(filePath));
    return new String(bytes);
}
}
```

Crea una classe che racchiuda le operazioni di dimensionamento automatico e Amazon EC2.

```
public class AutoScaler {

    private static Ec2Client ec2Client;
    private static AutoScalingClient autoScalingClient;
    private static IamClient iamClient;

    private static SsmClient ssmClient;

    private IamClient getIAMClient() {
        if (iamClient == null) {
            iamClient = IamClient.builder()
                .region(Region.US_EAST_1)
                .build();
        }
        return iamClient;
    }

    private SsmClient getSSMClient() {
        if (ssmClient == null) {
            ssmClient = SsmClient.builder()
                .region(Region.US_EAST_1)
                .build();
        }
        return ssmClient;
    }

    private Ec2Client getEc2Client() {
        if (ec2Client == null) {
```

```
        ec2Client = Ec2Client.builder()
            .region(Region.US_EAST_1)
            .build();
    }
    return ec2Client;
}

private AutoScalingClient getAutoScalingClient() {
    if (autoScalingClient == null) {
        autoScalingClient = AutoScalingClient.builder()
            .region(Region.US_EAST_1)
            .build();
    }
    return autoScalingClient;
}

/**
 * Terminates and instances in an EC2 Auto Scaling group. After an instance is
 * terminated, it can no longer be accessed.
 */
public void terminateInstance(String instanceId) {
    TerminateInstanceInAutoScalingGroupRequest terminateInstanceRequest =
    TerminateInstanceInAutoScalingGroupRequest
        .builder()
        .instanceId(instanceId)
        .shouldDecrementDesiredCapacity(false)
        .build();

    getAutoScalingClient().terminateInstanceInAutoScalingGroup(terminateInstanceRequest);
    System.out.format("Terminated instance %s.", instanceId);
}

/**
 * Replaces the profile associated with a running instance. After the profile is
 * replaced, the instance is rebooted to ensure that it uses the new profile.
 * When
 * the instance is ready, Systems Manager is used to restart the Python web
 * server.
 */
public void replaceInstanceProfile(String instanceId, String
newInstanceProfileName, String profileAssociationId)
    throws InterruptedException {
    // Create an IAM instance profile specification.
```

```
software.amazon.awssdk.services.ec2.model.IamInstanceProfileSpecification
iamInstanceProfile =
software.amazon.awssdk.services.ec2.model.IamInstanceProfileSpecification
    .builder()
    .name(newInstanceProfileName) // Make sure 'newInstanceProfileName'
is a valid IAM Instance Profile
                                // name.
    .build();

// Replace the IAM instance profile association for the EC2 instance.
ReplaceIamInstanceProfileAssociationRequest replaceRequest =
ReplaceIamInstanceProfileAssociationRequest
    .builder()
    .iamInstanceProfile(iamInstanceProfile)
    .associationId(profileAssociationId) // Make sure
'profileAssociationId' is a valid association ID.
    .build();

try {
    getEc2Client().replaceIamInstanceProfileAssociation(replaceRequest);
    // Handle the response as needed.
} catch (Ec2Exception e) {
    // Handle exceptions, log, or report the error.
    System.err.println("Error: " + e.getMessage());
}

System.out.format("Replaced instance profile for association %s with profile
%s.", profileAssociationId,
    newInstanceProfileName);
TimeUnit.SECONDS.sleep(15);
boolean instReady = false;
int tries = 0;

// Reboot after 60 seconds
while (!instReady) {
    if (tries % 6 == 0) {
        getEc2Client().rebootInstances(RebootInstancesRequest.builder()
            .instanceIds(instanceId)
            .build());
        System.out.println("Rebooting instance " + instanceId + " and
waiting for it to be ready.");
    }
    tries++;
    try {
        TimeUnit.SECONDS.sleep(10);
    }
}
```

```

        } catch (InterruptedException e) {
            e.printStackTrace();
        }

        DescribeInstanceInformationResponse informationResponse =
getSSMClient().describeInstanceInformation();
        List<InstanceInformation> instanceInformationList =
informationResponse.getInstanceInformationList();
        for (InstanceInformation info : instanceInformationList) {
            if (info.getInstanceId().equals(instanceId)) {
                instReady = true;
                break;
            }
        }
    }

    SendCommandRequest sendCommandRequest = SendCommandRequest.builder()
        .instanceIds(instanceId)
        .documentName("AWS-RunShellScript")
        .parameters(Collections.singletonMap("commands",
80")))
        .build();

    getSSMClient().sendCommand(sendCommandRequest);
    System.out.println("Restarted the Python web server on instance " +
instanceId + ".");
}

public void openInboundPort(String secGroupId, String port, String ipAddress) {
    AuthorizeSecurityGroupIngressRequest ingressRequest =
AuthorizeSecurityGroupIngressRequest.builder()
        .groupName(secGroupId)
        .cidrIp(ipAddress)
        .fromPort(Integer.parseInt(port))
        .build();

    getEc2Client().authorizeSecurityGroupIngress(ingressRequest);
    System.out.format("Authorized ingress to %s on port %s from %s.",
secGroupId, port, ipAddress);
}

/**
 * Detaches a role from an instance profile, detaches policies from the role,

```



```
    * and deletes all the resources.
    */
    public void deleteInstanceProfile(String roleName, String profileName) {
        try {
            software.amazon.awssdk.services.iam.model.GetInstanceProfileRequest
getInstanceProfileRequest =
software.amazon.awssdk.services.iam.model.GetInstanceProfileRequest
                .builder()
                .instanceProfileName(profileName)
                .build();

            GetInstanceProfileResponse response =
getIAMClient().getInstanceProfile(getInstanceProfileRequest);
            String name = response.instanceProfile().instanceProfileName();
            System.out.println(name);

            RemoveRoleFromInstanceProfileRequest profileRequest =
RemoveRoleFromInstanceProfileRequest.builder()
                .instanceProfileName(profileName)
                .roleName(roleName)
                .build();

            getIAMClient().removeRoleFromInstanceProfile(profileRequest);
            DeleteInstanceProfileRequest deleteInstanceProfileRequest =
DeleteInstanceProfileRequest.builder()
                .instanceProfileName(profileName)
                .build();

            getIAMClient().deleteInstanceProfile(deleteInstanceProfileRequest);
            System.out.println("Deleted instance profile " + profileName);

            DeleteRoleRequest deleteRoleRequest = DeleteRoleRequest.builder()
                .roleName(roleName)
                .build();

            // List attached role policies.
            ListAttachedRolePoliciesResponse rolesResponse = getIAMClient()
                .listAttachedRolePolicies(role -> role.roleName(roleName));
            List<AttachedPolicy> attachedPolicies =
rolesResponse.attachedPolicies();
            for (AttachedPolicy attachedPolicy : attachedPolicies) {
                DetachRolePolicyRequest request = DetachRolePolicyRequest.builder()
                    .roleName(roleName)
                    .policyArn(attachedPolicy.policyArn())
```

```
        .build();

        getIAMClient().detachRolePolicy(request);
        System.out.println("Detached and deleted policy " +
attachedPolicy.policyName());
    }

    getIAMClient().deleteRole(deleteRoleRequest);
    System.out.println("Instance profile and role deleted.");

} catch (IamException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
}

public void deleteTemplate(String templateName) {
    getEc2Client().deleteLaunchTemplate(name ->
name.launchTemplateName(templateName));
    System.out.format(templateName + " was deleted.");
}

public void deleteAutoScaleGroup(String groupName) {
    DeleteAutoScalingGroupRequest deleteAutoScalingGroupRequest =
DeleteAutoScalingGroupRequest.builder()
        .autoScalingGroupName(groupName)
        .forceDelete(true)
        .build();

getAutoScalingClient().deleteAutoScalingGroup(deleteAutoScalingGroupRequest);
    System.out.println(groupName + " was deleted.");
}

/*
 * Verify the default security group of the specified VPC allows ingress from
 * this
 * computer. This can be done by allowing ingress from this computer's IP
 * address. In some situations, such as connecting from a corporate network, you
 * must instead specify a prefix list ID. You can also temporarily open the port
 * to
 * any IP address while running this example. If you do, be sure to remove
 * public
 * access when you're done.
 */
```

```
*
*/
public GroupInfo verifyInboundPort(String VPC, int port, String ipAddress) {
    boolean portIsOpen = false;
    GroupInfo groupInfo = new GroupInfo();
    try {
        Filter filter = Filter.builder()
            .name("group-name")
            .values("default")
            .build();

        Filter filter1 = Filter.builder()
            .name("vpc-id")
            .values(VPC)
            .build();

        DescribeSecurityGroupsRequest securityGroupsRequest =
DescribeSecurityGroupsRequest.builder()
            .filters(filter, filter1)
            .build();

        DescribeSecurityGroupsResponse securityGroupsResponse = getEc2Client()
            .describeSecurityGroups(securityGroupsRequest);
        String securityGroup =
securityGroupsResponse.securityGroups().get(0).groupName();
        groupInfo.setGroupName(securityGroup);

        for (SecurityGroup secGroup : securityGroupsResponse.securityGroups()) {
            System.out.println("Found security group: " + secGroup.groupId());

            for (IpPermission ipPermission : secGroup.ipPermissions()) {
                if (ipPermission.fromPort() == port) {
                    System.out.println("Found inbound rule: " + ipPermission);
                    for (IpRange ipRange : ipPermission.ipRanges()) {
                        String cidrIp = ipRange.cidrIp();
                        if (cidrIp.startsWith(ipAddress) ||
cidrIp.equals("0.0.0.0/0")) {
                            System.out.println(cidrIp + " is applicable");
                            portIsOpen = true;
                        }
                    }
                }

                if (!ipPermission.prefixListIds().isEmpty()) {
                    System.out.println("Prefix lList is applicable");
                }
            }
        }
    }
}
```

```
        portIsOpen = true;
    }

    if (!portIsOpen) {
        System.out
            .println("The inbound rule does not appear to be
open to either this computer's IP,"
                    + " all IP addresses (0.0.0.0/0), or to
a prefix list ID.");
    } else {
        break;
    }
}
}

} catch (AutoScalingException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
}

groupInfo.setPortOpen(portIsOpen);
return groupInfo;
}

/**
 * Attaches an Elastic Load Balancing (ELB) target group to this EC2 Auto
 * Scaling group.
 * The target group specifies how the load balancer forward requests to the
 * instances
 * in the group.
 */
public void attachLoadBalancerTargetGroup(String asGroupName, String
targetGroupARN) {
    try {
        AttachLoadBalancerTargetGroupsRequest targetGroupsRequest =
AttachLoadBalancerTargetGroupsRequest.builder()
            .autoScalingGroupName(asGroupName)
            .targetGroupARNs(targetGroupARN)
            .build();

getAutoScalingClient().attachLoadBalancerTargetGroups(targetGroupsRequest);
        System.out.println("Attached load balancer to " + asGroupName);
    }
}
```

```
    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

// Creates an EC2 Auto Scaling group with the specified size.
public String[] createGroup(int groupSize, String templateName, String
autoScalingGroupName) {

    // Get availability zones.
    software.amazon.awssdk.services.ec2.model.DescribeAvailabilityZonesRequest
zonesRequest =
software.amazon.awssdk.services.ec2.model.DescribeAvailabilityZonesRequest
    .builder()
    .build();

    DescribeAvailabilityZonesResponse zonesResponse =
getEc2Client().describeAvailabilityZones(zonesRequest);
    List<String> availabilityZoneNames =
zonesResponse.availabilityZones().stream()

.map(software.amazon.awssdk.services.ec2.model.AvailabilityZone::zoneName)
    .collect(Collectors.toList());

    String availabilityZones = String.join(",", availabilityZoneNames);
    LaunchTemplateSpecification specification =
LaunchTemplateSpecification.builder()
    .launchTemplateName(templateName)
    .version("$Default")
    .build();

    String[] zones = availabilityZones.split(",");
    CreateAutoScalingGroupRequest groupRequest =
CreateAutoScalingGroupRequest.builder()
    .launchTemplate(specification)
    .availabilityZones(zones)
    .maxSize(groupSize)
    .minSize(groupSize)
    .autoScalingGroupName(autoScalingGroupName)
    .build();

    try {
        getAutoScalingClient().createAutoScalingGroup(groupRequest);
    }
```

```
    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    System.out.println("Created an EC2 Auto Scaling group named " +
autoScalingGroupName);
    return zones;
}

public String getDefaultVPC() {
    // Define the filter.
    Filter defaultFilter = Filter.builder()
        .name("is-default")
        .values("true")
        .build();

    software.amazon.awssdk.services.ec2.model.DescribeVpcsRequest request =
software.amazon.awssdk.services.ec2.model.DescribeVpcsRequest
        .builder()
        .filters(defaultFilter)
        .build();

    DescribeVpcsResponse response = getEc2Client().describeVpcs(request);
    return response.vpcs().get(0).vpcId();
}

// Gets the default subnets in a VPC for a specified list of Availability Zones.
public List<Subnet> getSubnets(String vpcId, String[] availabilityZones) {
    List<Subnet> subnets = null;
    Filter vpcFilter = Filter.builder()
        .name("vpc-id")
        .values(vpcId)
        .build();

    Filter azFilter = Filter.builder()
        .name("availability-zone")
        .values(availabilityZones)
        .build();

    Filter defaultForAZ = Filter.builder()
        .name("default-for-az")
        .values("true")
        .build();
```

```
DescribeSubnetsRequest request = DescribeSubnetsRequest.builder()
    .filters(vpcFilter, azFilter, defaultForAZ)
    .build();

DescribeSubnetsResponse response = getEc2Client().describeSubnets(request);
subnets = response.subnets();
return subnets;
}

// Gets data about the instances in the EC2 Auto Scaling group.
public String getBadInstance(String groupName) {
    DescribeAutoScalingGroupsRequest request =
DescribeAutoScalingGroupsRequest.builder()
    .autoScalingGroupNames(groupName)
    .build();

    DescribeAutoScalingGroupsResponse response =
getAutoScalingClient().describeAutoScalingGroups(request);
    AutoScalingGroup autoScalingGroup = response.autoScalingGroups().get(0);
    List<String> instanceIds = autoScalingGroup.instances().stream()
        .map(instance -> instance.instanceId())
        .collect(Collectors.toList());

    String[] instanceIdArray = instanceIds.toArray(new String[0]);
    for (String instanceId : instanceIdArray) {
        System.out.println("Instance ID: " + instanceId);
        return instanceId;
    }
    return "";
}

// Gets data about the profile associated with an instance.
public String getInstanceProfile(String instanceId) {
    Filter filter = Filter.builder()
        .name("instance-id")
        .values(instanceId)
        .build();

    DescribeIamInstanceProfileAssociationsRequest associationsRequest =
DescribeIamInstanceProfileAssociationsRequest
        .builder()
        .filters(filter)
        .build();
```

```

        DescribeIamInstanceProfileAssociationsResponse response = getEc2Client()
            .describeIamInstanceProfileAssociations(associationsRequest);
        return response.iamInstanceProfileAssociations().get(0).associationId();
    }

    public void deleteRolesPolicies(String policyName, String roleName, String
InstanceProfile) {
        ListPoliciesRequest listPoliciesRequest =
ListPoliciesRequest.builder().build();
        ListPoliciesResponse listPoliciesResponse =
getIAMClient().listPolicies(listPoliciesRequest);
        for (Policy policy : listPoliciesResponse.policies()) {
            if (policy.policyName().equals(policyName)) {
                // List the entities (users, groups, roles) that are attached to the
policy.

software.amazon.awssdk.services.iam.model.ListEntitiesForPolicyRequest
listEntitiesRequest =
software.amazon.awssdk.services.iam.model.ListEntitiesForPolicyRequest
                .builder()
                .policyArn(policy.arn())
                .build();
                ListEntitiesForPolicyResponse listEntitiesResponse = iamClient
                    .listEntitiesForPolicy(listEntitiesRequest);
                if (!listEntitiesResponse.policyGroups().isEmpty() || !
listEntitiesResponse.policyUsers().isEmpty()
                    || !listEntitiesResponse.policyRoles().isEmpty()) {
                    // Detach the policy from any entities it is attached to.
                    DetachRolePolicyRequest detachPolicyRequest =
DetachRolePolicyRequest.builder()
                        .policyArn(policy.arn())
                        .roleName(roleName) // Specify the name of the IAM role
                        .build();

                    getIAMClient().detachRolePolicy(detachPolicyRequest);
                    System.out.println("Policy detached from entities.");
                }

                // Now, you can delete the policy.
                DeletePolicyRequest deletePolicyRequest =
DeletePolicyRequest.builder()
                    .policyArn(policy.arn())
                    .build();

```



```
        getIAMClient().deletePolicy(deletePolicyRequest);
        System.out.println("Policy deleted successfully.");
        break;
    }
}

// List the roles associated with the instance profile
ListInstanceProfilesForRoleRequest listRolesRequest =
ListInstanceProfilesForRoleRequest.builder()
    .roleName(roleName)
    .build();

// Detach the roles from the instance profile
ListInstanceProfilesForRoleResponse listRolesResponse =
iamClient.listInstanceProfilesForRole(listRolesRequest);
for (software.amazon.awssdk.services.iam.model.InstanceProfile profile :
listRolesResponse.instanceProfiles()) {
    RemoveRoleFromInstanceProfileRequest removeRoleRequest =
RemoveRoleFromInstanceProfileRequest.builder()
        .instanceProfileName(InstanceProfile)
        .roleName(roleName) // Remove the extra dot here
        .build();

    getIAMClient().removeRoleFromInstanceProfile(removeRoleRequest);
    System.out.println("Role " + roleName + " removed from instance profile
" + InstanceProfile);
}

// Delete the instance profile after removing all roles
DeleteInstanceProfileRequest deleteInstanceProfileRequest =
DeleteInstanceProfileRequest.builder()
    .instanceProfileName(InstanceProfile)
    .build();

getIAMClient().deleteInstanceProfile(r ->
r.instanceProfileName(InstanceProfile));
System.out.println(InstanceProfile + " Deleted");
System.out.println("All roles and policies are deleted.");
}
}
```

## Crea una classe che racchiuda le operazioni di Elastic Load Balancing.

```
public class LoadBalancer {
    public ElasticLoadBalancingV2Client elasticLoadBalancingV2Client;

    public ElasticLoadBalancingV2Client getLoadBalancerClient() {
        if (elasticLoadBalancingV2Client == null) {
            elasticLoadBalancingV2Client = ElasticLoadBalancingV2Client.builder()
                .region(Region.US_EAST_1)
                .build();
        }

        return elasticLoadBalancingV2Client;
    }

    // Checks the health of the instances in the target group.
    public List<TargetHealthDescription> checkTargetHealth(String targetGroupName) {
        DescribeTargetGroupsRequest targetGroupsRequest =
        DescribeTargetGroupsRequest.builder()
            .names(targetGroupName)
            .build();

        DescribeTargetGroupsResponse tgResponse =
        getLoadBalancerClient().describeTargetGroups(targetGroupsRequest);

        DescribeTargetHealthRequest healthRequest =
        DescribeTargetHealthRequest.builder()
            .targetGroupArn(tgResponse.targetGroups().get(0).targetGroupArn())
            .build();

        DescribeTargetHealthResponse healthResponse =
        getLoadBalancerClient().describeTargetHealth(healthRequest);
        return healthResponse.targetHealthDescriptions();
    }

    // Gets the HTTP endpoint of the load balancer.
    public String getEndpoint(String lbName) {
        DescribeLoadBalancersResponse res = getLoadBalancerClient()
            .describeLoadBalancers(describe -> describe.names(lbName));
        return res.loadBalancers().get(0).dnsName();
    }

    // Deletes a load balancer.
    public void deleteLoadBalancer(String lbName) {
```

```

    try {
        // Use a waiter to delete the Load Balancer.
        DescribeLoadBalancersResponse res = getLoadBalancerClient()
            .describeLoadBalancers(describe -> describe.names(lbName));
        ElasticLoadBalancingV2Waiter loadBalancerWaiter =
getLoadBalancerClient().waiter();
        DescribeLoadBalancersRequest request =
DescribeLoadBalancersRequest.builder()
            .loadBalancerArns(res.loadBalancers().get(0).loadBalancerArn())
            .build();

        getLoadBalancerClient().deleteLoadBalancer(
            builder ->
builder.loadBalancerArn(res.loadBalancers().get(0).loadBalancerArn()));
        WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =
loadBalancerWaiter
            .waitUntilLoadBalancersDeleted(request);
        waiterResponse.matched().response().ifPresent(System.out::println);

    } catch (ElasticLoadBalancingV2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }
    System.out.println(lbName + " was deleted.");
}

// Deletes the target group.
public void deleteTargetGroup(String targetGroupName) {
    try {
        DescribeTargetGroupsResponse res = getLoadBalancerClient()
            .describeTargetGroups(describe ->
describe.names(targetGroupName));
        getLoadBalancerClient()
            .deleteTargetGroup(builder ->
builder.targetGroupArn(res.targetGroups().get(0).targetGroupArn()));
    } catch (ElasticLoadBalancingV2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }
    System.out.println(targetGroupName + " was deleted.");
}

// Verify this computer can successfully send a GET request to the load balancer
// endpoint.
public boolean verifyLoadBalancerEndpoint(String elbDnsName) throws IOException,
InterruptedException {

```

```
boolean success = false;
int retries = 3;
CloseableHttpClient httpClient = HttpClients.createDefault();

// Create an HTTP GET request to the ELB.
HttpGet httpGet = new HttpGet("http://" + elbDnsName);
try {
    while ((!success) && (retries > 0)) {
        // Execute the request and get the response.
        HttpResponse response = httpClient.execute(httpGet);
        int statusCode = response.getStatusLine().getStatusCode();
        System.out.println("HTTP Status Code: " + statusCode);
        if (statusCode == 200) {
            success = true;
        } else {
            retries--;
            System.out.println("Got connection error from load balancer
endpoint, retrying...");
            TimeUnit.SECONDS.sleep(15);
        }
    }

} catch (org.apache.http.conn.HttpHostConnectException e) {
    System.out.println(e.getMessage());
}

System.out.println("Status.." + success);
return success;
}

/*
 * Creates an Elastic Load Balancing target group. The target group specifies
 * how
 * the load balancer forward requests to instances in the group and how instance
 * health is checked.
 */
public String createTargetGroup(String protocol, int port, String vpcId, String
targetGroupName) {
    CreateTargetGroupRequest targetGroupRequest =
CreateTargetGroupRequest.builder()
        .healthCheckPath("/healthcheck")
        .healthCheckTimeoutSeconds(5)
        .port(port)
        .vpcId(vpcId)
```

```
        .name(targetGroupName)
        .protocol(protocol)
        .build();

        CreateTargetGroupResponse targetGroupResponse =
getLoadBalancerClient().createTargetGroup(targetGroupRequest);
        String targetGroupArn =
targetGroupResponse.targetGroups().get(0).targetGroupArn();
        String targetGroup =
targetGroupResponse.targetGroups().get(0).targetGroupName();
        System.out.println("The " + targetGroup + " was created with ARN" +
targetGroupArn);
        return targetGroupArn;
    }

    /**
     * Creates an Elastic Load Balancing load balancer that uses the specified
     * subnets
     * and forwards requests to the specified target group.
     */
    public String createLoadBalancer(List<Subnet> subnetIds, String targetGroupARN,
String lbName, int port,
        String protocol) {
        try {
            List<String> subnetIdStrings = subnetIds.stream()
                .map(Subnet::subnetId)
                .collect(Collectors.toList());

            CreateLoadBalancerRequest balancerRequest =
CreateLoadBalancerRequest.builder()
                .subnets(subnetIdStrings)
                .name(lbName)
                .scheme("internet-facing")
                .build();

            // Create and wait for the load balancer to become available.
            CreateLoadBalancerResponse lsResponse =
getLoadBalancerClient().createLoadBalancer(balancerRequest);
            String lbARN = lsResponse.loadBalancers().get(0).loadBalancerArn();

            ElasticLoadBalancingV2Waiter loadBalancerWaiter =
getLoadBalancerClient().waiter();
            DescribeLoadBalancersRequest request =
DescribeLoadBalancersRequest.builder()
```

```
        .loadBalancerArns(lbARN)
        .build();

        System.out.println("Waiting for Load Balancer " + lbName + " to become
available.");
        WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =
loadBalancerWaiter
            .waitUntilLoadBalancerAvailable(request);
        waiterResponse.matched().response().ifPresent(System.out::println);
        System.out.println("Load Balancer " + lbName + " is available.");

        // Get the DNS name (endpoint) of the load balancer.
        String lbDNSName = lsResponse.loadBalancers().get(0).dnsName();
        System.out.println("*** Load Balancer DNS Name: " + lbDNSName);

        // Create a listener for the load balance.
        Action action = Action.builder()
            .targetGroupArn(targetGroupARN)
            .type("forward")
            .build();

        CreateListenerRequest listenerRequest = CreateListenerRequest.builder()

.loadBalancerArn(lsResponse.loadBalancers().get(0).loadBalancerArn())
            .defaultActions(action)
            .port(port)
            .protocol(protocol)
            .defaultActions(action)
            .build();

        getLoadBalancerClient().createListener(listenerRequest);
        System.out.println("Created listener to forward traffic from load
balancer " + lbName + " to target group "
            + targetGroupARN);

        // Return the load balancer DNS name.
        return lbDNSName;

    } catch (ElasticLoadBalancingV2Exception e) {
        e.printStackTrace();
    }
    return "";
}
}
```

Crea una classe che utilizzi DynamoDB per simulare un servizio di raccomandazione.

```
public class Database {

    private static DynamoDbClient dynamoDbClient;

    public static DynamoDbClient getDynamoDbClient() {
        if (dynamoDbClient == null) {
            dynamoDbClient = DynamoDbClient.builder()
                .region(Region.US_EAST_1)
                .build();
        }
        return dynamoDbClient;
    }

    // Checks to see if the Amazon DynamoDB table exists.
    private boolean doesTableExist(String tableName) {
        try {
            // Describe the table and catch any exceptions.
            DescribeTableRequest describeTableRequest =
DescribeTableRequest.builder()
                .tableName(tableName)
                .build();

            getDynamoDbClient().describeTable(describeTableRequest);
            System.out.println("Table '" + tableName + "' exists.");
            return true;

        } catch (ResourceNotFoundException e) {
            System.out.println("Table '" + tableName + "' does not exist.");
        } catch (DynamoDbException e) {
            System.err.println("Error checking table existence: " + e.getMessage());
        }
        return false;
    }

    /**
     * Creates a DynamoDB table to use a recommendation service. The table has a
     * hash key named 'MediaType' that defines the type of media recommended, such
     * as
     * Book or Movie, and a range key named 'ItemId' that, combined with the
```

```
* MediaType,
* forms a unique identifier for the recommended item.
*/
public void createTable(String tableName, String fileName) throws IOException {
    // First check to see if the table exists.
    boolean doesExist = doesTableExist(tableName);
    if (!doesExist) {
        DynamoDbWaiter dbWaiter = getDynamoDbClient().waiter();
        CreateTableRequest createTableRequest = CreateTableRequest.builder()
            .tableName(tableName)
            .attributeDefinitions(
                AttributeDefinition.builder()
                    .attributeName("MediaType")
                    .attributeType(ScalarAttributeType.S)
                    .build(),
                AttributeDefinition.builder()
                    .attributeName("ItemId")
                    .attributeType(ScalarAttributeType.N)
                    .build())
            .keySchema(
                KeySchemaElement.builder()
                    .attributeName("MediaType")
                    .keyType(KeyType.HASH)
                    .build(),
                KeySchemaElement.builder()
                    .attributeName("ItemId")
                    .keyType(KeyType.RANGE)
                    .build())
            .provisionedThroughput(
                ProvisionedThroughput.builder()
                    .readCapacityUnits(5L)
                    .writeCapacityUnits(5L)
                    .build())
            .build();

        getDynamoDbClient().createTable(createTableRequest);
        System.out.println("Creating table " + tableName + "...");

        // Wait until the Amazon DynamoDB table is created.
        DescribeTableRequest tableRequest = DescribeTableRequest.builder()
            .tableName(tableName)
            .build();
    }
}
```



```
        WaiterResponse<DescribeTableResponse> waiterResponse =
dbWaiter.waitUntilTableExists(tableRequest);
        waiterResponse.matched().response().ifPresent(System.out::println);
        System.out.println("Table " + tableName + " created.");

        // Add records to the table.
        populateTable(fileName, tableName);
    }
}

public void deleteTable(String tableName) {
    getDynamoDbClient().deleteTable(table -> table.tableName(tableName));
    System.out.println("Table " + tableName + " deleted.");
}

// Populates the table with data located in a JSON file using the DynamoDB
// enhanced client.
public void populateTable(String fileName, String tableName) throws IOException
{
    DynamoDbEnhancedClient enhancedClient = DynamoDbEnhancedClient.builder()
        .dynamoDbClient(getDynamoDbClient())
        .build();
    ObjectMapper objectMapper = new ObjectMapper();
    File jsonFile = new File(fileName);
    JsonNode rootNode = objectMapper.readTree(jsonFile);

    DynamoDbTable<Recommendation> mappedTable = enhancedClient.table(tableName,
        TableSchema.fromBean(Recommendation.class));
    for (JsonNode currentNode : rootNode) {
        String mediaType = currentNode.path("MediaType").path("S").asText();
        int itemId = currentNode.path("ItemId").path("N").asInt();
        String title = currentNode.path("Title").path("S").asText();
        String creator = currentNode.path("Creator").path("S").asText();

        // Create a Recommendation object and set its properties.
        Recommendation rec = new Recommendation();
        rec.setMediaType(mediaType);
        rec.setItemId(itemId);
        rec.setTitle(title);
        rec.setCreator(creator);

        // Put the item into the DynamoDB table.
        mappedTable.putItem(rec); // Add the Recommendation to the list.
    }
}
```

```
        System.out.println("Added all records to the " + tableName);
    }
}
```

Crea una classe che racchiuda le operazioni di Systems Manager.

```
public class ParameterHelper {

    String tableName = "doc-example-resilient-architecture-table";
    String dyntable = "doc-example-recommendation-service";
    String failureResponse = "doc-example-resilient-architecture-failure-response";
    String healthCheck = "doc-example-resilient-architecture-health-check";

    public void reset() {
        put(dyntable, tableName);
        put(failureResponse, "none");
        put(healthCheck, "shallow");
    }

    public void put(String name, String value) {
        SsmClient ssmClient = SsmClient.builder()
            .region(Region.US_EAST_1)
            .build();

        PutParameterRequest parameterRequest = PutParameterRequest.builder()
            .name(name)
            .value(value)
            .overwrite(true)
            .type("String")
            .build();

        ssmClient.putParameter(parameterRequest);
        System.out.printf("Setting demo parameter %s to '%s'.", name, value);
    }
}
```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella Documentazione di riferimento delle API AWS SDK for Java 2.x .
  - [AttachLoadBalancerTargetGroups](#)
  - [CreateAutoScalingGroup](#)

- [CreateInstanceProfile](#)
- [CreateLaunchTemplate](#)
- [CreateListener](#)
- [CreateLoadBalancer](#)
- [CreateTargetGroup](#)
- [DeleteAutoScalingGroup](#)
- [DeleteInstanceProfile](#)
- [DeleteLaunchTemplate](#)
- [DeleteLoadBalancer](#)
- [DeleteTargetGroup](#)
- [DescribeAutoScalingGroups](#)
- [DescribeAvailabilityZones](#)
- [DescribeIamInstanceProfileAssociations](#)
- [DescribeInstances](#)
- [DescribeLoadBalancers](#)
- [DescribeSubnets](#)
- [DescribeTargetGroups](#)
- [DescribeTargetHealth](#)
- [DescribeVpcs](#)
- [RebootInstances](#)
- [ReplaceIamInstanceProfileAssociation](#)
- [TerminateInstanceInAutoScalingGroup](#)
- [UpdateAutoScalingGroup](#)

Creazione di un utente e assunzione di un ruolo

Il seguente esempio di codice mostra come creare un utente e assumere un ruolo.

**⚠ Warning**

Per evitare rischi per la sicurezza, non utilizzare gli utenti IAM per l'autenticazione quando sviluppi software creato ad hoc o lavori con dati reali. Utilizza invece la federazione con un provider di identità come [AWS IAM Identity Center](#).

- Crea un utente che non disponga di autorizzazioni.
- Crea un ruolo che conceda l'autorizzazione per elencare i bucket Amazon S3 per l'account.
- Aggiungi una policy per consentire all'utente di assumere il ruolo.
- Assumi il ruolo ed elenca i bucket S3 utilizzando le credenziali temporanee, quindi ripulisci le risorse.

## SDK per Java 2.x

**i Note**

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Crea funzioni che eseguono il wrapping delle operazioni degli utenti IAM.

```
/*  
  To run this Java V2 code example, set up your development environment, including  
  your credentials.  
  
  For information, see this documentation topic:  
  
  https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html  
  
  This example performs these operations:  
  
  1. Creates a user that has no permissions.  
  2. Creates a role and policy that grants Amazon S3 permissions.  
  3. Creates a role.  
  4. Grants the user permissions.  
  5. Gets temporary credentials by assuming the role.  Creates an Amazon S3 Service  
  client object with the temporary credentials.
```

6. Deletes the resources.

\*/

```
public class IAMScenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");
    public static final String PolicyDocument = "{" +
        "  \"Version\": \"2012-10-17\", " +
        "  \"Statement\": [" +
        "    {" +
        "      \"Effect\": \"Allow\", " +
        "      \"Action\": [" +
        "        \"s3:*\" " +
        "      ], " +
        "      \"Resource\": \"*\\" " +
        "    } " +
        "  ] " +
        "}";

    public static String userArn;

    public static void main(String[] args) throws Exception {

        final String usage = ""

            Usage:
            <username> <policyName> <roleName> <roleSessionName>
<bucketName>\s

            Where:
            username - The name of the IAM user to create.\s
            policyName - The name of the policy to create.\s
            roleName - The name of the role to create.\s
            roleSessionName - The name of the session required for the
assumeRole operation.\s
            bucketName - The name of the Amazon S3 bucket from which objects
are read.\s

            """;

        if (args.length != 5) {
            System.out.println(usage);
            System.exit(1);
        }

        String userName = args[0];
```

```
String policyName = args[1];
String roleName = args[2];
String roleSessionName = args[3];
String bucketName = args[4];

Region region = Region.AWS_GLOBAL;
IamClient iam = IamClient.builder()
    .region(region)
    .build();

System.out.println(DASHES);
System.out.println("Welcome to the AWS IAM example scenario.");
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println(" 1. Create the IAM user.");
User createUser = createIAMUser(iam, userName);

System.out.println(DASHES);
userArn = createUser.arn();

AccessKey myKey = createIAMAccessKey(iam, userName);
String accessKey = myKey.accessKeyId();
String secretKey = myKey.secretAccessKey();
String assumeRolePolicyDocument = "{" +
    "\"Version\": \"2012-10-17\"," +
    "\"Statement\": [{" +
    "\"Effect\": \"Allow\"," +
    "\"Principal\": {" +
    "  \"AWS\": \"\" + userArn + "\"" +
    "}," +
    "\"Action\": \"sts:AssumeRole\"" +
    "}]}" +
    "}";

System.out.println(assumeRolePolicyDocument);
System.out.println(userName + " was successfully created.");
System.out.println(DASHES);
System.out.println("2. Creates a policy.");
String polArn = createIAMPolicy(iam, policyName);
System.out.println("The policy " + polArn + " was successfully created.");
System.out.println(DASHES);

System.out.println(DASHES);
```

```
System.out.println("3. Creates a role.");
TimeUnit.SECONDS.sleep(30);
String roleArn = createIAMRole(iam, roleName, assumeRolePolicyDocument);
System.out.println(roleArn + " was successfully created.");
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("4. Grants the user permissions.");
attachIAMRolePolicy(iam, roleName, polArn);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("*** Wait for 30 secs so the resource is available");
TimeUnit.SECONDS.sleep(30);
System.out.println("5. Gets temporary credentials by assuming the role.");
System.out.println("Perform an Amazon S3 Service operation using the
temporary credentials.");
assumeRole(roleArn, roleSessionName, bucketName, accessKey, secretKey);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("6 Getting ready to delete the AWS resources");
deleteKey(iam, userName, accessKey);
deleteRole(iam, roleName, polArn);
deleteIAMUser(iam, userName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("This IAM Scenario has successfully completed");
System.out.println(DASHES);
}

public static AccessKey createIAMAccessKey(IamClient iam, String user) {
    try {
        CreateAccessKeyRequest request = CreateAccessKeyRequest.builder()
            .userName(user)
            .build();

        CreateAccessKeyResponse response = iam.createAccessKey(request);
        return response.accessKey();
    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

```
    }
    return null;
}

public static User createIAMUser(IamClient iam, String username) {
    try {
        // Create an IamWaiter object
        IamWaiter iamWaiter = iam.waiter();
        CreateUserRequest request = CreateUserRequest.builder()
            .userName(username)
            .build();

        // Wait until the user is created.
        CreateUserResponse response = iam.createUser(request);
        GetUserRequest userRequest = GetUserRequest.builder()
            .userName(response.user().userName())
            .build();

        WaiterResponse<GetUserResponse> waitUntilUserExists =
iamWaiter.waitUntilUserExists(userRequest);
        waitUntilUserExists.matched().response().ifPresent(System.out::println);
        return response.user();

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return null;
}

public static String createIAMRole(IamClient iam, String rolename, String json)
{

    try {
        CreateRoleRequest request = CreateRoleRequest.builder()
            .roleName(rolename)
            .assumeRolePolicyDocument(json)
            .description("Created using the AWS SDK for Java")
            .build();

        CreateRoleResponse response = iam.createRole(request);
        System.out.println("The ARN of the role is " + response.role().arn());
        return response.role().arn();
    }
}
```



```
    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}

public static String createIAMPolicy(IamClient iam, String policyName) {
    try {
        // Create an IamWaiter object.
        IamWaiter iamWaiter = iam.waiter();
        CreatePolicyRequest request = CreatePolicyRequest.builder()
            .policyName(policyName)
            .policyDocument(PolicyDocument).build();

        CreatePolicyResponse response = iam.createPolicy(request);
        GetPolicyRequest polRequest = GetPolicyRequest.builder()
            .policyArn(response.policy().arn())
            .build();

        WaiterResponse<GetPolicyResponse> waitUntilPolicyExists =
iamWaiter.waitUntilPolicyExists(polRequest);

waitUntilPolicyExists.matched().response().ifPresent(System.out::println);
        return response.policy().arn();

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}

public static void attachIAMRolePolicy(IamClient iam, String roleName, String
policyArn) {
    try {
        ListAttachedRolePoliciesRequest request =
ListAttachedRolePoliciesRequest.builder()
            .roleName(roleName)
            .build();

        ListAttachedRolePoliciesResponse response =
iam.listAttachedRolePolicies(request);
        List<AttachedPolicy> attachedPolicies = response.attachedPolicies();
    }
}
```

```
        String polArn;
        for (AttachedPolicy policy : attachedPolicies) {
            polArn = policy.policyArn();
            if (polArn.compareTo(policyArn) == 0) {
                System.out.println(roleName + " policy is already attached to
this role.");
                return;
            }
        }

        AttachRolePolicyRequest attachRequest =
AttachRolePolicyRequest.builder()
            .roleName(roleName)
            .policyArn(policyArn)
            .build();

        iam.attachRolePolicy(attachRequest);
        System.out.println("Successfully attached policy " + policyArn + " to
role " + roleName);

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

// Invoke an Amazon S3 operation using the Assumed Role.
public static void assumeRole(String roleArn, String roleSessionName, String
bucketName, String keyVal,
    String keySecret) {

    // Use the creds of the new IAM user that was created in this code example.
    AwsBasicCredentials credentials = AwsBasicCredentials.create(keyVal,
keySecret);
    StsClient stsClient = StsClient.builder()
        .region(Region.US_EAST_1)
        .credentialsProvider(StaticCredentialsProvider.create(credentials))
        .build();

    try {
        AssumeRoleRequest roleRequest = AssumeRoleRequest.builder()
            .roleArn(roleArn)
            .roleSessionName(roleSessionName)
            .build();
```

```
AssumeRoleResponse roleResponse = stsClient.assumeRole(roleRequest);
Credentials myCreds = roleResponse.credentials();
String key = myCreds.accessKeyId();
String secKey = myCreds.secretAccessKey();
String secToken = myCreds.sessionToken();

// List all objects in an Amazon S3 bucket using the temp creds
retrieved by
// invoking assumeRole.
Region region = Region.US_EAST_1;
S3Client s3 = S3Client.builder()
    .credentialsProvider(
StaticCredentialsProvider.create(AwsSessionCredentials.create(key, secKey,
secToken)))
    .region(region)
    .build();

System.out.println("Created a S3Client using temp credentials.");
System.out.println("Listing objects in " + bucketName);
ListObjectsRequest listObjects = ListObjectsRequest.builder()
    .bucket(bucketName)
    .build();

ListObjectsResponse res = s3.listObjects(listObjects);
List<S3Object> objects = res.contents();
for (S3Object myValue : objects) {
    System.out.println("The name of the key is " + myValue.key());
    System.out.println("The owner is " + myValue.owner());
}

} catch (StsException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
}

public static void deleteRole(IamClient iam, String roleName, String polArn) {

    try {
        // First the policy needs to be detached.
        DetachRolePolicyRequest rolePolicyRequest =
DetachRolePolicyRequest.builder()
```

```
        .policyArn(polArn)
        .roleName(roleName)
        .build();

iam.detachRolePolicy(rolePolicyRequest);

// Delete the policy.
DeletePolicyRequest request = DeletePolicyRequest.builder()
    .policyArn(polArn)
    .build();

iam.deletePolicy(request);
System.out.println("*** Successfully deleted " + polArn);

// Delete the role.
DeleteRoleRequest roleRequest = DeleteRoleRequest.builder()
    .roleName(roleName)
    .build();

iam.deleteRole(roleRequest);
System.out.println("*** Successfully deleted " + roleName);

} catch (IamException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}

public static void deleteKey(IamClient iam, String username, String accessKey) {
    try {
        DeleteAccessKeyRequest request = DeleteAccessKeyRequest.builder()
            .accessKeyId(accessKey)
            .userName(username)
            .build();

        iam.deleteAccessKey(request);
        System.out.println("Successfully deleted access key " + accessKey +
            " from user " + username);

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

```
public static void deleteIAMUser(IamClient iam, String userName) {
    try {
        DeleteUserRequest request = DeleteUserRequest.builder()
            .userName(userName)
            .build();

        iam.deleteUser(request);
        System.out.println("*** Successfully deleted " + userName);

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella Documentazione di riferimento delle API AWS SDK for Java 2.x .
  - [AttachRolePolicy](#)
  - [CreateAccessKey](#)
  - [CreatePolicy](#)
  - [CreateRole](#)
  - [CreateUser](#)
  - [DeleteAccessKey](#)
  - [DeletePolicy](#)
  - [DeleteRole](#)
  - [DeleteUser](#)
  - [DeleteUserPolicy](#)
  - [DetachRolePolicy](#)
  - [PutUserPolicy](#)

Lavora con l'API IAM Policy Builder

L'esempio di codice seguente mostra come:

- Crea policy IAM utilizzando l'API orientata agli oggetti.
- Usa l'API IAM Policy Builder con il servizio IAM.

## SDK per Java 2.x

### Note

C'è dell'altro GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Gli esempi utilizzano le seguenti importazioni.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.policybuilder.iam.IamConditionOperator;
import software.amazon.awssdk.policybuilder.iam.IamEffect;
import software.amazon.awssdk.policybuilder.iam.IamPolicy;
import software.amazon.awssdk.policybuilder.iam.IamPolicyWriter;
import software.amazon.awssdk.policybuilder.iam.IamPrincipal;
import software.amazon.awssdk.policybuilder.iam.IamPrincipalType;
import software.amazon.awssdk.policybuilder.iam.IamResource;
import software.amazon.awssdk.policybuilder.iam.IamStatement;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.GetPolicyResponse;
import software.amazon.awssdk.services.iam.model.GetPolicyVersionResponse;
import software.amazon.awssdk.services.sts.StsClient;

import java.net.URLDecoder;
import java.nio.charset.StandardCharsets;
import java.util.Arrays;
import java.util.List;
```

Crea una policy basata sul tempo.

```
public String timeBasedPolicyExample() {
    IamPolicy policy = IamPolicy.builder()
```

```

        .addStatement(b -> b
            .effect(IamEffect.ALLOW)
            .addAction("dynamodb:GetItem")
            .addResource(IamResource.ALL)
            .addCondition(b1 -> b1

.operator(IamConditionOperator.DATE_GREATER_THAN)

.key("aws:CurrentTime")

.value("2020-04-01T00:00:00Z"))

            .addCondition(b1 -> b1

.operator(IamConditionOperator.DATE_LESS_THAN)

.key("aws:CurrentTime")

.value("2020-06-30T23:59:59Z")))
        .build();

// Use an IamPolicyWriter to write out the JSON string to a more
readable
// format.
return policy.toJson(IamPolicyWriter.builder()
    .prettyPrint(true)
    .build());
}

```

Crea una policy con più condizioni.

```

public String multipleConditionsExample() {
    IamPolicy policy = IamPolicy.builder()
        .addStatement(b -> b
            .effect(IamEffect.ALLOW)
            .addAction("dynamodb:GetItem")
            .addAction("dynamodb:BatchGetItem")
            .addAction("dynamodb:Query")
            .addAction("dynamodb:PutItem")
            .addAction("dynamodb:UpdateItem")
            .addAction("dynamodb:DeleteItem")

.addAction("dynamodb:BatchWriteItem")

```

```

.addResource("arn:aws:dynamodb:*:*:table/table-name")

.addConditions(IamConditionOperator.STRING_EQUALS

.addPrefix("ForAllValues:"),

"dynamodb:Attributes",

List.of("column-
name1", "column-name2", "column-name3"))

.addCondition(b1 -> b1

.operator(IamConditionOperator.STRING_EQUALS

.addSuffix("IfExists"))

.key("dynamodb:Select")

.value("SPECIFIC_ATTRIBUTES"))

.build();

return policy.toJson(IamPolicyWriter.builder()
.prettyPrint(true).build());
}

```

Usa i principi in una policy.

```

public String specifyPrincipalsExample() {
    IamPolicy policy = IamPolicy.builder()
        .addStatement(b -> b
            .effect(IamEffect.DENY)
            .addAction("s3:*")
            .addPrincipal(IamPrincipal.ALL)

.addResource("arn:aws:s3:::BUCKETNAME/*")

.addResource("arn:aws:s3:::BUCKETNAME")

.addCondition(b1 -> b1

.operator(IamConditionOperator.ARN_NOT_EQUALS)

.key("aws:PrincipalArn")

```



```

        .value("arn:aws:iam::444455556666:user/user-name")))
            .build();
        return policy.toJson(IamPolicyWriter.builder()
            .prettyPrint(true).build());
    }

```

### Consentire l'accesso multi-account .

```

    public String allowCrossAccountAccessExample() {
        IamPolicy policy = IamPolicy.builder()
            .addStatement(b -> b
                .effect(IamEffect.ALLOW)
                .addPrincipal(IamPrincipalType.AWS,
                    "111122223333")
                .addAction("s3:PutObject")
                .addResource("arn:aws:s3:::DOC-
                    EXAMPLE-BUCKET/*")
                .addCondition(b1 -> b1
                    .operator(IamConditionOperator.STRING_EQUALS)
                    .key("s3:x-amz-acl")
                    .value("bucket-
                    owner-full-control")))
            .build();
        return policy.toJson(IamPolicyWriter.builder()
            .prettyPrint(true).build());
    }

```

### Crea e carica un IamPolicy.

```

    public String createAndUploadPolicyExample(IamClient iam, String accountID,
        String policyName) {
        // Build the policy.
        IamPolicy policy = IamPolicy.builder() // 'version' defaults to
            "2012-10-17".
            .addStatement(IamStatement.builder()
                .effect(IamEffect.ALLOW)
                .addAction("dynamodb:PutItem")
                .addResource("arn:aws:dynamodb:us-
                    east-1:" + accountID

```

```

        + ":table/"
exampleTableName")
        .build()
        .build();
// Upload the policy.
iam.createPolicy(r ->
r.policyName(policyName).policyDocument(policy.toJson()));
return
policy.toJson(IamPolicyWriter.builder().prettyPrint(true).build());
}

```

### Scarica e lavora con un IamPolicy.

```

public String createNewBasedOnExistingPolicyExample(IamClient iam, String
accountID, String policyName,
String newPolicyName) {

String policyArn = "arn:aws:iam::" + accountID + ":policy/" +
policyName;
GetPolicyResponse getPolicyResponse = iam.getPolicy(r ->
r.policyArn(policyArn));

String policyVersion =
getPolicyResponse.policy().defaultVersionId();
GetPolicyVersionResponse getPolicyVersionResponse = iam
.getPolicyVersion(r ->
r.policyArn(policyArn).versionId(policyVersion));

// Create an IamPolicy instance from the JSON string returned from
IAM.
String decodedPolicy =
URLDecoder.decode(getPolicyVersionResponse.policyVersion().document(),
StandardCharsets.UTF_8);
IamPolicy policy = IamPolicy.fromJson(decodedPolicy);

/*
* All IamPolicy components are immutable, so use the copy method
that creates a
* new instance that
* can be altered in the same method call.
*/
}

```

```
        * Add the ability to get an item from DynamoDB as an additional
action.
        */
        iamStatement newStatement = policy.statements().get(0).copy(s ->
s.addAction("dynamodb:GetItem"));

        // Create a new statement that replaces the original statement.
        iamPolicy newPolicy = policy.copy(p ->
p.statements(Arrays.asList(newStatement)));

        // Upload the new policy. IAM now has both policies.
        iam.createPolicy(r -> r.policyName(newPolicyName)
                .policyDocument(newPolicy.toJson()));

        return
newPolicy.toJson(iamPolicyWriter.builder().prettyPrint(true).build());
    }
}
```

- Per ulteriori informazioni, consulta la [Guida per sviluppatori di AWS SDK for Java 2.x](#).
- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella Documentazione di riferimento delle API AWS SDK for Java 2.x .
  - [CreatePolicy](#)
  - [GetPolicy](#)
  - [GetPolicyVersion](#)

## AWS IoT esempi che utilizzano SDK for Java 2.x

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando AWS SDK for Java 2.x with AWS IoT.

Le operazioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le operazioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati e negli esempi tra servizi.

Scenari: esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio.

Ogni esempio include un collegamento a GitHub, dove è possibile trovare istruzioni su come configurare ed eseguire il codice nel contesto.

## Nozioni di base

### Salve AWS IoT

L'esempio di codice seguente mostra come iniziare a utilizzare AWS IoT.

### SDK per Java 2.x

#### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iot.IotClient;
import software.amazon.awssdk.services.iot.model.ListThingsRequest;
import software.amazon.awssdk.services.iot.model.ListThingsResponse;
import software.amazon.awssdk.services.iot.model.ThingAttribute;
import java.util.List;

public class HelloIoT {
    public static void main(String[] args) {
        System.out.println("Hello AWS IoT. Here is a listing of your AWS IoT
Things:");
        IotClient iotClient = IotClient.builder()
            .region(Region.US_EAST_1)
            .build();

        listAllThings(iotClient);
    }

    public static void listAllThings( IotClient iotClient) {
        ListThingsRequest thingsRequest = ListThingsRequest.builder()
            .maxResults(10)
            .build();

        ListThingsResponse response = iotClient.listThings(thingsRequest) ;
        List<ThingAttribute> thingList = response.things();
        for (ThingAttribute attribute : thingList) {
            System.out.println("Thing name: "+attribute.thingName());
            System.out.println("Thing ARN: "+attribute.thingArn());
        }
    }
}
```

```
    }  
  }  
}
```

- Per i dettagli sull'API, consulta [ListThings](#) in AWS SDK for Java 2.x API Reference.

## Argomenti

- [Azioni](#)
- [Scenari](#)

## Azioni

### AttachThingPrincipal

Il seguente esempio di codice mostra come utilizzare `AttachThingPrincipal`.

#### SDK per Java 2.x

#### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public static void attachCertificateToThing(IotClient iotClient, String  
thingName, String certificateArn) {  
    // Attach the certificate to the thing.  
    AttachThingPrincipalRequest principalRequest =  
AttachThingPrincipalRequest.builder()  
        .thingName(thingName)  
        .principal(certificateArn)  
        .build();  
  
    AttachThingPrincipalResponse attachResponse =  
iotClient.attachThingPrincipal(principalRequest);  
  
    // Verify the attachment was successful.  
    if (attachResponse.sdkHttpResponse().isSuccessful()) {
```

```
        System.out.println("Certificate attached to Thing successfully.");

        // Print additional information about the Thing.
        describeThing(iotClient, thingName);
    } else {
        System.err.println("Failed to attach certificate to Thing. HTTP Status
Code: " +
            attachResponse.sdkHttpResponse().statusCode());
    }
}
```

- Per i dettagli sull'API, [AttachThingPrincipal](#) consulta AWS SDK for Java 2.x API Reference.

## CreateKeysAndCertificate

Il seguente esempio di codice mostra come utilizzare `CreateKeysAndCertificate`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public static String createCertificate(IotClient iotClient) {
    try {
        CreateKeysAndCertificateResponse response =
iotClient.createKeysAndCertificate();
        String certificatePem = response.certificatePem();
        String certificateArn = response.certificateArn();

        // Print the details.
        System.out.println("\nCertificate:");
        System.out.println(certificatePem);
        System.out.println("\nCertificate ARN:");
        System.out.println(certificateArn);
        return certificateArn;
    } catch (IotException e) {
```

```
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }

    return "";
}
```

- Per i dettagli sull'API, [CreateKeysAndCertificate](#) consulta AWS SDK for Java 2.x API Reference.

## CreateThing

Il seguente esempio di codice mostra come utilizzare `CreateThing`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public static void createIoTThing(IotClient iotClient, String thingName) {
    try {
        CreateThingRequest createThingRequest = CreateThingRequest.builder()
            .thingName(thingName)
            .build();

        CreateThingResponse createThingResponse =
iotClient.createThing(createThingRequest);
        System.out.println(thingName + " was successfully created. The ARN value
is " + createThingResponse.thingArn());

    } catch (IotException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Per i dettagli sull'API, [CreateThing](#) consulta AWS SDK for Java 2.x API Reference.

## CreateTopicRule

Il seguente esempio di codice mostra come utilizzare `CreateTopicRule`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public static void createIoTRule(IotClient iotClient, String roleARN, String
ruleName, String action) {
    try {
        String sql = "SELECT * FROM '" + TOPIC + "'";
        SnsAction action1 = SnsAction.builder()
            .targetArn(action)
            .roleArn(roleARN)
            .build();

        // Create the action.
        Action myAction = Action.builder()
            .sns(action1)
            .build();

        // Create the topic rule payload.
        TopicRulePayload topicRulePayload = TopicRulePayload.builder()
            .sql(sql)
            .actions(myAction)
            .build();

        // Create the topic rule request.
        CreateTopicRuleRequest topicRuleRequest =
CreateTopicRuleRequest.builder()
            .ruleName(ruleName)
            .topicRulePayload(topicRulePayload)
            .build();

        // Create the rule.
        iotClient.createTopicRule(topicRuleRequest);
        System.out.println("IoT Rule created successfully.");
    }
}
```



```
    } catch (IotException e) {  
        System.err.println(e.getMessage());  
        System.exit(1);  
    }  
}
```

- Per i dettagli sull'API, [CreateTopicRule](#) consulta AWS SDK for Java 2.x API Reference.

## DeleteCertificate

Il seguente esempio di codice mostra come utilizzare `DeleteCertificate`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).


```
public static void deleteCertificate(IotClient iotClient, String  
certificateArn ) {  
    DeleteCertificateRequest certificateProviderRequest =  
DeleteCertificateRequest.builder()  
        .certificateId(extractCertificateId(certificateArn))  
        .build();  
  
    iotClient.deleteCertificate(certificateProviderRequest);  
    System.out.println(certificateArn + " was successfully deleted.");  
}
```

- Per i dettagli sull'API, [DeleteCertificate](#) consulta AWS SDK for Java 2.x API Reference.

## DeleteThing

Il seguente esempio di codice mostra come utilizzare `DeleteThing`.

## SDK per Java 2.x

 Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public static void deleteIoTThing(IotClient iotClient, String thingName) {
    try {
        DeleteThingRequest deleteThingRequest = DeleteThingRequest.builder()
            .thingName(thingName)
            .build();

        iotClient.deleteThing(deleteThingRequest);
        System.out.println("Deleted Thing " + thingName);


    } catch (IotException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Per i dettagli sull'API, [DeleteThing](#) consulta AWS SDK for Java 2.x API Reference.

## DescribeEndpoint

Il seguente esempio di codice mostra come utilizzare `DescribeEndpoint`.

## SDK per Java 2.x

 Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public static String describeEndpoint(IotClient iotClient) {
```

```
    try {
        DescribeEndpointResponse endpointResponse =
iotClient.describeEndpoint(DescribeEndpointRequest.builder().build());

        // Get the endpoint URL.
        String endpointUrl = endpointResponse.endpointAddress();
        String exString = getValue(endpointUrl);
        String fullEndpoint = "https://" + exString + "-ats.iot.us-
east-1.amazonaws.com";

        System.out.println("Full Endpoint URL: " + fullEndpoint);
        return fullEndpoint;

    } catch (IotException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "" ;
}
```

- Per i dettagli sull'API, [DescribeEndpoint](#) consulta AWS SDK for Java 2.x API Reference.

## DescribeThing

Il seguente esempio di codice mostra come utilizzare `DescribeThing`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
private static void describeThing(IotClient iotClient, String thingName) {
    try {
        DescribeThingRequest thingRequest = DescribeThingRequest.builder()
            .thingName(thingName)
            .build() ;
    }
```

```
// Print Thing details.
DescribeThingResponse describeResponse =
iotClient.describeThing(thingRequest);
System.out.println("Thing Details:");
System.out.println("Thing Name: " + describeResponse.thingName());
System.out.println("Thing ARN: " + describeResponse.thingArn());

} catch (IotException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

- Per i dettagli sull'API, [DescribeThing](#) consulta AWS SDK for Java 2.x API Reference.

## DetachThingPrincipal

Il seguente esempio di codice mostra come utilizzare `DetachThingPrincipal`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public static void detachThingPrincipal(IotClient iotClient, String thingName,
String certificateArn){
    try {
        DetachThingPrincipalRequest thingPrincipalRequest =
        DetachThingPrincipalRequest.builder()
            .principal(certificateArn)
            .thingName(thingName)
            .build();

        iotClient.detachThingPrincipal(thingPrincipalRequest);
        System.out.println(certificateArn + " was successfully removed from "
+thingName);
    }
}
```

```
    } catch (IotException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

- Per i dettagli sull'API, [DetachThingPrincipal](#) consulta AWS SDK for Java 2.x API Reference.

## ListCertificates

Il seguente esempio di codice mostra come utilizzare `ListCertificates`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).


```
public static void listCertificates(IotClient iotClient) {  
    ListCertificatesResponse response = iotClient.listCertificates();  
    List<Certificate> certList = response.certificates();  
    for (Certificate cert : certList) {  
        System.out.println("Cert id: " + cert.certificateId());  
        System.out.println("Cert Arn: " + cert.certificateArn());  
    }  
}
```

- Per i dettagli sull'API, [ListCertificates](#) consulta AWS SDK for Java 2.x API Reference.

## SearchIndex

Il seguente esempio di codice mostra come utilizzare `SearchIndex`.

## SDK per Java 2.x

 Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public static void searchThings(IotClient iotClient, String queryString){
    SearchIndexRequest searchIndexRequest = SearchIndexRequest.builder()
        .queryString(queryString)
        .build();

    try {
        // Perform the search and get the result.
        SearchIndexResponse searchIndexResponse =
        iotClient.searchIndex(searchIndexRequest);

        // Process the result.
        if (searchIndexResponse.things().isEmpty()) {
            System.out.println("No things found.");
        } else {
            searchIndexResponse.things().forEach(thing ->
            System.out.println("Thing id found using search is " + thing.thingId()));
        }
    } catch (IotException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Per i dettagli sull'API, [SearchIndex](#) consulta AWS SDK for Java 2.x API Reference.

## UpdateThing

Il seguente esempio di codice mostra come utilizzare `UpdateThing`.

## SDK per Java 2.x

 Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public static void updateThing(IotClient iotClient, String thingName) {
    // Specify the new attribute values.
    String newLocation = "Office";
    String newFirmwareVersion = "v2.0";

    Map<String, String> attMap = new HashMap<>();
    attMap.put("location", newLocation);
    attMap.put("firmwareVersion", newFirmwareVersion);

    AttributePayload attributePayload = AttributePayload.builder()
        .attributes(attMap)
        .build();

    UpdateThingRequest updateThingRequest = UpdateThingRequest.builder()
        .thingName(thingName)
        .attributePayload(attributePayload)
        .build();

    try {
        // Update the IoT Thing attributes.
        iotClient.updateThing(updateThingRequest);
        System.out.println("Thing attributes updated successfully.");
    } catch (IotException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Per i dettagli sull'API, [UpdateThing](#) consulta AWS SDK for Java 2.x API Reference.

## Scenari

Lavora con i casi d'uso della gestione dei dispositivi

Il seguente esempio di codice mostra come utilizzare i casi d'uso della gestione dei AWS IoT dispositivi utilizzando AWS IoT SDK

SDK per Java 2.x

### Note

C'è altro su [GitHub](#) Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iot.IotClient;
import software.amazon.awssdk.services.iot.model.Action;
import software.amazon.awssdk.services.iot.model.AttachThingPrincipalRequest;
import software.amazon.awssdk.services.iot.model.AttachThingPrincipalResponse;
import software.amazon.awssdk.services.iot.model.AttributePayload;
import software.amazon.awssdk.services.iot.model.Certificate;
import software.amazon.awssdk.services.iot.model.CreateKeysAndCertificateResponse;
import software.amazon.awssdk.services.iot.model.CreateThingRequest;
import software.amazon.awssdk.services.iot.model.CreateTopicRuleRequest;
import software.amazon.awssdk.services.iot.model.DeleteCertificateRequest;
import software.amazon.awssdk.services.iot.model.CreateThingResponse;
import software.amazon.awssdk.services.iot.model.DeleteThingRequest;
import software.amazon.awssdk.services.iot.model.DescribeEndpointRequest;
import software.amazon.awssdk.services.iot.model.DescribeEndpointResponse;
import software.amazon.awssdk.services.iot.model.DescribeThingRequest;
import software.amazon.awssdk.services.iot.model.DescribeThingResponse;
import software.amazon.awssdk.services.iot.model.DetachThingPrincipalRequest;
import software.amazon.awssdk.services.iot.model.IotException;
import software.amazon.awssdk.services.iot.model.ListCertificatesResponse;
import software.amazon.awssdk.services.iot.model.ListTopicRulesRequest;
import software.amazon.awssdk.services.iot.model.ListTopicRulesResponse;
import software.amazon.awssdk.services.iot.model.SearchIndexRequest;
import software.amazon.awssdk.services.iot.model.SearchIndexResponse;
import software.amazon.awssdk.services.iot.model.SnsAction;
import software.amazon.awssdk.services.iot.model.TopicRuleListItem;
```



```
import software.amazon.awssdk.services.iot.model.TopicRulePayload;
import software.amazon.awssdk.services.iot.model.UpdateThingRequest;
import software.amazon.awssdk.services.iotdataplane.IotDataPlaneClient;
import software.amazon.awssdk.services.iotdataplane.model.GetThingShadowRequest;
import software.amazon.awssdk.services.iotdataplane.model.GetThingShadowResponse;
import software.amazon.awssdk.services.iotdataplane.model.UpdateThingShadowRequest;
import java.net.URI;
import java.nio.charset.StandardCharsets;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.Scanner;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * This Java example performs these tasks:
 *
 * 1. Creates an AWS IoT Thing.
 * 2. Generate and attach a device certificate.
 * 3. Update an AWS IoT Thing with Attributes.
 * 4. Get an AWS IoT Endpoint.
 * 5. List your certificates.
 * 6. Updates the shadow for the specified thing..
 * 7. Write out the state information, in JSON format
 * 8. Creates a rule
 * 9. List rules
 * 10. Search things
 * 11. Detach and delete the certificate.
 * 12. Delete Thing.
 */
public class IotScenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");
    private static final String TOPIC = "your-iot-topic";
    public static void main(String[] args) {
        final String usage =
            ""
```

```
Usage:
    <roleARN> <snsAction>

Where:
    roleARN - The ARN of an IAM role that has permission to work
with AWS IOT.
    snsAction - An ARN of an SNS topic.
""";

if (args.length != 2) {
    System.out.println(usage);
    System.exit(1);
}

String thingName;
String ruleName;
String roleARN = args[0];
String snsAction = args[1];
Scanner scanner = new Scanner(System.in);
IoTClient iotClient = IoTClient.builder()
    .region(Region.US_EAST_1)
    .build();

System.out.println(DASHES);
System.out.println("Welcome to the AWS IoT example workflow.");
System.out.println("""
    This example program demonstrates various interactions with the AWS
Internet of Things (IoT) Core service. The program guides you through a series of
steps,
    including creating an IoT Thing, generating a device certificate,
updating the Thing with attributes, and so on.
    It utilizes the AWS SDK for Java V2 and incorporates functionality for
creating and managing IoT Things, certificates, rules,
    shadows, and performing searches. The program aims to showcase AWS IoT
capabilities and provides a comprehensive example for
    developers working with AWS IoT in a Java environment.

""");
System.out.print("Press Enter to continue...");
scanner.nextLine();
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("1. Create an AWS IoT Thing.");
```

```
System.out.println("""
    An AWS IoT Thing represents a virtual entity in the AWS IoT service that
    can be associated with a physical device.
    """);
// Prompt the user for input.
System.out.print("Enter Thing name: ");
thingName = scanner.nextLine();
createIoTThing(iotClient, thingName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("2. Generate a device certificate.");
System.out.println("""
    A device certificate performs a role in securing the communication
    between devices (Things) and the AWS IoT platform.
    """);

System.out.print("Do you want to create a certificate for " +thingName +"?
(y/n)");
String certAns = scanner.nextLine();
String certificateArn="" ;
if (certAns != null && certAns.trim().equalsIgnoreCase("y")) {
    certificateArn = createCertificate(iotClient);
    System.out.println("Attach the certificate to the AWS IoT Thing.");
    attachCertificateToThing(iotClient, thingName, certificateArn);
} else {
    System.out.println("A device certificate was not created.");
}
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("3. Update an AWS IoT Thing with Attributes.");
System.out.println("""
    IoT Thing attributes, represented as key-value pairs, offer a pivotal
    advantage in facilitating efficient data
    management and retrieval within the AWS IoT ecosystem.
    """);
System.out.print("Press Enter to continue...");
scanner.nextLine();
updateThing(iotClient, thingName);
System.out.println(DASHES);

System.out.println(DASHES);
```

```
System.out.println("4. Return a unique endpoint specific to the Amazon Web
Services account.");
System.out.println("""
    An IoT Endpoint refers to a specific URL or Uniform Resource Locator
that serves as the entry point for communication between IoT devices and the AWS
IoT service.
    """);
System.out.print("Press Enter to continue...");
scanner.nextLine();
String endpointUrl = describeEndpoint(iotClient);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("5. List your AWS IoT certificates");
System.out.print("Press Enter to continue...");
scanner.nextLine();
if (certificateArn.length() > 0) {
    listCertificates(iotClient);
} else {
    System.out.println("You did not create a certificates. Skipping this
step.");
}
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("6. Create an IoT shadow that refers to a digital
representation or virtual twin of a physical IoT device");
System.out.println("""
    A Thing Shadow refers to a feature that enables you to create a virtual
representation, or "shadow,"
    of a physical device or thing. The Thing Shadow allows you to
synchronize and control the state of a device between
    the cloud and the device itself. and the AWS IoT service. For example,
you can write and retrieve JSON data from a Thing Shadow.
    """);
System.out.print("Press Enter to continue...");
scanner.nextLine();
IotDataPlaneClient iotPlaneClient = IotDataPlaneClient.builder()
    .region(Region.US_EAST_1)
    .endpointOverride(URI.create(endpointUrl))
    .build();

updateShadowThing(iotPlaneClient, thingName);
System.out.println(DASHES);
```

```
System.out.println(DASHES);
System.out.println("7. Write out the state information, in JSON format.");
System.out.print("Press Enter to continue...");
scanner.nextLine();
getPayload(iotPlaneClient, thingName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("8. Creates a rule");
System.out.println("""
Creates a rule that is an administrator-level action.
Any user who has permission to create rules will be able to access data
processed by the rule.
""");
System.out.print("Enter Rule name: ");
ruleName = scanner.nextLine();
createIoTRule(iotClient, roleARN, ruleName, snsAction);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("9. List your rules.");
System.out.print("Press Enter to continue...");
scanner.nextLine();
listIoTRules(iotClient);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("10. Search things using the Thing name.");
System.out.print("Press Enter to continue...");
scanner.nextLine();
String queryString = "thingName:"+thingName ;
searchThings(iotClient, queryString);
System.out.println(DASHES);

System.out.println(DASHES);
if (certificateArn.length() > 0) {
    System.out.print("Do you want to detach and delete the certificate for "
+thingName +"? (y/n)");
    String delAns = scanner.nextLine();
    if (delAns != null && delAns.trim().equalsIgnoreCase("y")) {
        System.out.println("11. You selected to detach amd delete the
certificate.");
        System.out.print("Press Enter to continue...");
```

```
        scanner.nextLine();
        detachThingPrincipal(iotClient, thingName, certificateArn);
        deleteCertificate(iotClient, certificateArn);
    } else {
        System.out.println("11. You selected not to delete the
certificate.");
    }
    } else {
        System.out.println("11. You did not create a certificate so there is
nothing to delete.");
    }
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("12. Delete the AWS IoT Thing.");
    System.out.print("Do you want to delete the IoT Thing? (y/n)");
    String delAns = scanner.nextLine();
    if (delAns != null && delAns.trim().equalsIgnoreCase("y")) {
        deleteIoTThing(iotClient, thingName);
    } else {
        System.out.println("The IoT Thing was not deleted.");
    }
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("The AWS IoT workflow has successfully completed.");
    System.out.println(DASHES);
}

public static void listCertificates(IotClient iotClient) {
    ListCertificatesResponse response = iotClient.listCertificates();
    List<Certificate> certList = response.certificates();
    for (Certificate cert : certList) {
        System.out.println("Cert id: " + cert.certificateId());
        System.out.println("Cert Arn: " + cert.certificateArn());
    }
}

public static void listIoTRules(IotClient iotClient) {
    try {
        ListTopicRulesRequest listTopicRulesRequest =
ListTopicRulesRequest.builder().build();
        ListTopicRulesResponse listTopicRulesResponse =
iotClient.listTopicRules(listTopicRulesRequest);
    }
```

```
        System.out.println("List of IoT Rules:");
        List<TopicRuleListItem> ruleList = listTopicRulesResponse.rules();
        for (TopicRuleListItem rule : ruleList) {
            System.out.println("Rule Name: " + rule.ruleName());
            System.out.println("Rule ARN: " + rule.ruleArn());
            System.out.println("-----");
        }

    } catch (IotException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void createIoTRule(IotClient iotClient, String roleARN, String
ruleName, String action) {
    try {
        String sql = "SELECT * FROM '" + TOPIC + "'";
        SnsAction action1 = SnsAction.builder()
            .targetArn(action)
            .roleArn(roleARN)
            .build();

        // Create the action.
        Action myAction = Action.builder()
            .sns(action1)
            .build();

        // Create the topic rule payload.
        TopicRulePayload topicRulePayload = TopicRulePayload.builder()
            .sql(sql)
            .actions(myAction)
            .build();

        // Create the topic rule request.
        CreateTopicRuleRequest topicRuleRequest =
CreateTopicRuleRequest.builder()
            .ruleName(ruleName)
            .topicRulePayload(topicRulePayload)
            .build();

        // Create the rule.
        iotClient.createTopicRule(topicRuleRequest);
        System.out.println("IoT Rule created successfully.");
    }
}
```

```
    } catch (IotException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void getPayload(IotDataPlaneClient iotPlaneClient, String
thingName) {
    try {
        GetThingShadowRequest getThingShadowRequest =
GetThingShadowRequest.builder()
            .thingName(thingName)
            .build();

        GetThingShadowResponse getThingShadowResponse =
iotPlaneClient.getThingShadow(getThingShadowRequest);

        // Extracting payload from response.
        SdkBytes payload = getThingShadowResponse.payload();
        String payloadString = payload.asUtf8String();
        System.out.println("Received Shadow Data: " + payloadString);

    } catch (IotException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void updateShadowThing(IotDataPlaneClient iotPlaneClient, String
thingName) {
    try {
        // Create Thing Shadow State Document.
        String stateDocument = "{\"state\":{\"reported\":{\"temperature\":25,
\"humidity\":50}}}\"";
        SdkBytes data= SdkBytes.fromString(stateDocument,
StandardCharsets.UTF_8 );
        UpdateThingShadowRequest updateThingShadowRequest =
UpdateThingShadowRequest.builder()
            .thingName(thingName)
            .payload(data)
            .build();

        // Update Thing Shadow.
```



```
        iotPlaneClient.updateThingShadow(updateThingShadowRequest);
        System.out.println("Thing Shadow updated successfully.");

    } catch (IotException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void updateThing(IotClient iotClient, String thingName) {
    // Specify the new attribute values.
    String newLocation = "Office";
    String newFirmwareVersion = "v2.0";

    Map<String, String> attMap = new HashMap<>();
    attMap.put("location", newLocation);
    attMap.put("firmwareVersion", newFirmwareVersion);

    AttributePayload attributePayload = AttributePayload.builder()
        .attributes(attMap)
        .build();

    UpdateThingRequest updateThingRequest = UpdateThingRequest.builder()
        .thingName(thingName)
        .attributePayload(attributePayload)
        .build();

    try {
        // Update the IoT Thing attributes.
        iotClient.updateThing(updateThingRequest);
        System.out.println("Thing attributes updated successfully.");

    } catch (IotException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static String describeEndpoint(IotClient iotClient) {
    try {
        DescribeEndpointResponse endpointResponse =
        iotClient.describeEndpoint(DescribeEndpointRequest.builder().build());

        // Get the endpoint URL.
```

```
        String endpointUrl = endpointResponse.endpointAddress();
        String exString = getValue(endpointUrl);
        String fullEndpoint = "https://" + exString + "-ats.iot.us-
east-1.amazonaws.com";

        System.out.println("Full Endpoint URL: " + fullEndpoint);
        return fullEndpoint;

    } catch (IotException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "" ;
}

public static void detachThingPrincipal(IotClient iotClient, String thingName,
String certificateArn){
    try {
        DetachThingPrincipalRequest thingPrincipalRequest =
DetachThingPrincipalRequest.builder()
            .principal(certificateArn)
            .thingName(thingName)
            .build();

        iotClient.detachThingPrincipal(thingPrincipalRequest);
        System.out.println(certificateArn + " was successfully removed from "
+thingName);

    } catch (IotException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void deleteCertificate(IotClient iotClient, String
certificateArn ) {
    DeleteCertificateRequest certificateProviderRequest =
DeleteCertificateRequest.builder()
        .certificateId(extractCertificateId(certificateArn))
        .build();

    iotClient.deleteCertificate(certificateProviderRequest);
    System.out.println(certificateArn + " was successfully deleted.");
}
```

```
// Get the cert Id from the Cert ARN value.
private static String extractCertificateId(String certificateArn) {
    // Example ARN: arn:aws:iot:region:account-id:cert/certificate-id.
    String[] arnParts = certificateArn.split(":");
    String certificateIdPart = arnParts[arnParts.length - 1];
    return certificateIdPart.substring(certificateIdPart.lastIndexOf("/") + 1);
}

public static String createCertificate(IotClient iotClient) {
    try {
        CreateKeysAndCertificateResponse response =
iotClient.createKeysAndCertificate();
        String certificatePem = response.certificatePem();
        String certificateArn = response.certificateArn();

        // Print the details.
        System.out.println("\nCertificate:");
        System.out.println(certificatePem);
        System.out.println("\nCertificate ARN:");
        System.out.println(certificateArn);
        return certificateArn;

    } catch (IotException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }

    return "";
}

public static void attachCertificateToThing(IotClient iotClient, String
thingName, String certificateArn) {
    // Attach the certificate to the thing.
    AttachThingPrincipalRequest principalRequest =
AttachThingPrincipalRequest.builder()
        .thingName(thingName)
        .principal(certificateArn)
        .build();

    AttachThingPrincipalResponse attachResponse =
iotClient.attachThingPrincipal(principalRequest);

    // Verify the attachment was successful.
```

```
        if (attachResponse.sdkHttpResponse().isSuccessful()) {
            System.out.println("Certificate attached to Thing successfully.");

            // Print additional information about the Thing.
            describeThing(iotClient, thingName);
        } else {
            System.err.println("Failed to attach certificate to Thing. HTTP Status
Code: " +
                attachResponse.sdkHttpResponse().statusCode());
        }
    }

    private static void describeThing(IotClient iotClient, String thingName) {
        try {
            DescribeThingRequest thingRequest = DescribeThingRequest.builder()
                .thingName(thingName)
                .build();

            // Print Thing details.
            DescribeThingResponse describeResponse =
iotClient.describeThing(thingRequest);
            System.out.println("Thing Details:");
            System.out.println("Thing Name: " + describeResponse.thingName());
            System.out.println("Thing ARN: " + describeResponse.thingArn());

        } catch (IotException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void deleteIoTThing(IotClient iotClient, String thingName) {
        try {
            DeleteThingRequest deleteThingRequest = DeleteThingRequest.builder()
                .thingName(thingName)
                .build();

            iotClient.deleteThing(deleteThingRequest);
            System.out.println("Deleted Thing " + thingName);

        } catch (IotException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

```
}

public static void createIoTThing(IotClient iotClient, String thingName) {
    try {
        CreateThingRequest createThingRequest = CreateThingRequest.builder()
            .thingName(thingName)
            .build();

        CreateThingResponse createThingResponse =
iotClient.createThing(createThingRequest);
        System.out.println(thingName + " was successfully created. The ARN value
is " + createThingResponse.thingArn());

    } catch (IotException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

private static String getValue(String input) {
    // Define a regular expression pattern for extracting the subdomain.
    Pattern pattern = Pattern.compile("^(.*?)\\.iot\\.us-east-1\\.amazonaws\\
\\.com");

    // Match the pattern against the input string.
    Matcher matcher = pattern.matcher(input);

    // Check if a match is found.
    if (matcher.find()) {
        // Extract the subdomain from the first capturing group.
        String subdomain = matcher.group(1);
        System.out.println("Extracted subdomain: " + subdomain);
        return subdomain ;
    } else {
        System.out.println("No match found");
    }
    return "" ;
}

public static void searchThings(IotClient iotClient, String queryString){
    SearchIndexRequest searchIndexRequest = SearchIndexRequest.builder()
        .queryString(queryString)
        .build();
```

```
    try {
        // Perform the search and get the result.
        SearchIndexResponse searchIndexResponse =
iotClient.searchIndex(searchIndexRequest);

        // Process the result.
        if (searchIndexResponse.things().isEmpty()) {
            System.out.println("No things found.");
        } else {
            searchIndexResponse.things().forEach(thing ->
System.out.println("Thing id found using search is " + thing.thingId()));
        }
    } catch (IotException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

## AWS IoT data esempi che utilizzano SDK for Java 2.x

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando AWS SDK for Java 2.x with AWS IoT data.

Le operazioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le operazioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati e negli esempi tra servizi.

Scenari: esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio.

Ogni esempio include un collegamento a GitHub, dove è possibile trovare istruzioni su come configurare ed eseguire il codice nel contesto.

### Argomenti

- [Azioni](#)

## Azioni

### GetThingShadow

Il seguente esempio di codice mostra come utilizzare `GetThingShadow`.

SDK per Java 2.x

#### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public static void getPayload(IotDataPlaneClient iotPlaneClient, String
thingName) {
    try {
        GetThingShadowRequest getThingShadowRequest =
GetThingShadowRequest.builder()
            .thingName(thingName)
            .build();

        GetThingShadowResponse getThingShadowResponse =
iotPlaneClient.getThingShadow(getThingShadowRequest);

        // Extracting payload from response.
        SdkBytes payload = getThingShadowResponse.payload();
        String payloadString = payload.asUtf8String();
        System.out.println("Received Shadow Data: " + payloadString);

    } catch (IotException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- Per i dettagli sull'API, [GetThingShadow](#) consulta AWS SDK for Java 2.x API Reference.

### UpdateThingShadow

Il seguente esempio di codice mostra come utilizzare `UpdateThingShadow`.

## SDK per Java 2.x

 Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public static void updateShadowThing(IotDataPlaneClient iotPlaneClient, String
thingName) {
    try {
        // Create Thing Shadow State Document.
        String stateDocument = "{\"state\":{\"reported\":{\"temperature\":25,
        \"humidity\":50}}}\"";
        SdkBytes data= SdkBytes.fromString(stateDocument,
StandardCharsets.UTF_8 );
        UpdateThingShadowRequest updateThingShadowRequest =
UpdateThingShadowRequest.builder()
            .thingName(thingName)
            .payload(data)
            .build();

        // Update Thing Shadow.
        iotPlaneClient.updateThingShadow(updateThingShadowRequest);
        System.out.println("Thing Shadow updated successfully.");

    } catch (IotException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Per i dettagli sull'API, [UpdateThingShadow](#) consulta AWS SDK for Java 2.x API Reference.

## Esempi di Amazon Keyspaces con SDK for Java 2.x

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando AWS SDK for Java 2.x con Amazon Keyspaces.



Le operazioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le operazioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati e negli esempi tra servizi.

Scenari: esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio.

Ogni esempio include un collegamento a GitHub, dove puoi trovare istruzioni su come configurare ed eseguire il codice nel contesto.

## Nozioni di base

### Salve Amazon Keyspaces

I seguenti esempi di codice mostrano come iniziare a utilizzare Amazon Keyspaces.

## SDK per Java 2.x

### Note

C'è altro su [GitHub](#) Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.keyspaces.KeyspacesClient;
import software.amazon.awssdk.services.keyspaces.model.KeyspaceSummary;
import software.amazon.awssdk.services.keyspaces.model.KeyspacesException;
import software.amazon.awssdk.services.keyspaces.model.ListKeyspacesRequest;
import software.amazon.awssdk.services.keyspaces.model.ListKeyspacesResponse;
import java.util.List;

/**
 * Before running this Java (v2) code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class HelloKeyspaces {
    public static void main(String[] args) {
```

```
    Region region = Region.US_EAST_1;
    KeyspacesClient keyClient = KeyspacesClient.builder()
        .region(region)
        .build();

    listKeyspaces(keyClient);
}

public static void listKeyspaces(KeyspacesClient keyClient) {
    try {
        ListKeyspacesRequest keyspacesRequest = ListKeyspacesRequest.builder()
            .maxResults(10)
            .build();

        ListKeyspacesResponse response =
keyClient.listKeyspaces(keyspacesRequest);
        List<KeyspaceSummary> keyspaces = response.keyspaces();
        for (KeyspaceSummary keyspace : keyspaces) {
            System.out.println("The name of the keyspace is " +
keyspace.keyspaceName());
        }

    } catch (KeyspacesException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Per i dettagli sull'API, [ListKeyspaces](#) consulta AWS SDK for Java 2.x API Reference.

## Argomenti

- [Azioni](#)
- [Scenari](#)

## Azioni

### CreateKeyspace

Il seguente esempio di codice mostra come utilizzare `CreateKeyspace`.

## SDK per Java 2.x

 Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public static void createKeySpace(KeyspacesClient keyClient, String
keyspaceName) {
    try {
        CreateKeyspaceRequest keyspaceRequest = CreateKeyspaceRequest.builder()
            .keyspaceName(keyspaceName)
            .build();

        CreateKeyspaceResponse response =
keyClient.createKeyspace(keyspaceRequest);
        System.out.println("The ARN of the KeySpace is " +
response.resourceArn());


    } catch (KeyspacesException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Per i dettagli sull'API, [CreateKeyspace](#) consulta AWS SDK for Java 2.x API Reference.

## CreateTable

Il seguente esempio di codice mostra come utilizzare CreateTable.

## SDK per Java 2.x

 Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public static void createTable(KeyspacesClient keyClient, String keySpace,
String tableName) {
    try {
        // Set the columns.
        ColumnDefinition defTitle = ColumnDefinition.builder()
            .name("title")
            .type("text")
            .build();

        ColumnDefinition defYear = ColumnDefinition.builder()
            .name("year")
            .type("int")
            .build();

        ColumnDefinition defReleaseDate = ColumnDefinition.builder()
            .name("release_date")
            .type("timestamp")
            .build();

        ColumnDefinition defPlot = ColumnDefinition.builder()
            .name("plot")
            .type("text")
            .build();

        List<ColumnDefinition> colList = new ArrayList<>();
        colList.add(defTitle);
        colList.add(defYear);
        colList.add(defReleaseDate);
        colList.add(defPlot);

        // Set the keys.
        PartitionKey yearKey = PartitionKey.builder()
            .name("year")
            .build();

        PartitionKey titleKey = PartitionKey.builder()
            .name("title")
            .build();

        List<PartitionKey> keyList = new ArrayList<>();
        keyList.add(yearKey);
        keyList.add(titleKey);
    }
}
```

```

        SchemaDefinition schemaDefinition = SchemaDefinition.builder()
            .partitionKeys(keyList)
            .allColumns(colList)
            .build();

        PointInTimeRecovery timeRecovery = PointInTimeRecovery.builder()
            .status(PointInTimeRecoveryStatus.ENABLED)
            .build();

        CreateTableRequest tableRequest = CreateTableRequest.builder()
            .keyspaceName(keySpace)
            .tableName(tableName)
            .schemaDefinition(schemaDefinition)
            .pointInTimeRecovery(timeRecovery)
            .build();

        CreateTableResponse response = keyClient.createTable(tableRequest);
        System.out.println("The table ARN is " + response.resourceArn());

    } catch (KeyspacesException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

```

- Per i dettagli sull'API, [CreateTable](#) consulta AWS SDK for Java 2.x API Reference.

## DeleteKeyspace

Il seguente esempio di codice mostra come utilizzare `DeleteKeyspace`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```

public static void deleteKeyspace(KeyspacesClient keyClient, String
keyspaceName) {

```

```
    try {
        DeleteKeyspaceRequest deleteKeyspaceRequest =
DeleteKeyspaceRequest.builder()
        .keyspaceName(keyspaceName)
        .build();

        keyClient.deleteKeyspace(deleteKeyspaceRequest);

    } catch (KeyspacesException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Per i dettagli sull'API, [DeleteKeyspace](#) consulta AWS SDK for Java 2.x API Reference.

## DeleteTable

Il seguente esempio di codice mostra come utilizzare DeleteTable.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public static void deleteTable(KeyspacesClient keyClient, String keyspaceName,
String tableName) {
    try {
        DeleteTableRequest tableRequest = DeleteTableRequest.builder()
        .keyspaceName(keyspaceName)
        .tableName(tableName)
        .build();

        keyClient.deleteTable(tableRequest);

    } catch (KeyspacesException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

```
    }  
}
```

- Per i dettagli sull'API, [DeleteTable](#) consulta AWS SDK for Java 2.x API Reference.

## GetKeyspace

Il seguente esempio di codice mostra come utilizzare `GetKeyspace`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public static void checkKeyspaceExistence(KeyspacesClient keyClient, String  
keyspaceName) {  
    try {  
        GetKeyspaceRequest keyspaceRequest = GetKeyspaceRequest.builder()  
            .keyspaceName(keyspaceName)  
            .build();  
  
        GetKeyspaceResponse response = keyClient.getKeyspace(keyspaceRequest);  
        String name = response.keyspaceName();  
        System.out.println("The " + name + " KeySpace is ready");  
  
    } catch (KeyspacesException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

- Per i dettagli sull'API, [GetKeyspace](#) consulta AWS SDK for Java 2.x API Reference.

## GetTable

Il seguente esempio di codice mostra come utilizzare `GetTable`.

## SDK per Java 2.x

 Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public static void checkTable(KeyspacesClient keyClient, String keyspaceName,
String tableName)
    throws InterruptedException {
    try {
        boolean tableStatus = false;
        String status;
        GetTableResponse response = null;
        GetTableRequest tableRequest = GetTableRequest.builder()
            .keyspaceName(keyspaceName)
            .tableName(tableName)
            .build();

        while (!tableStatus) {
            response = keyClient.getTable(tableRequest);
            status = response.statusAsString();
            System.out.println(". The table status is " + status);

            if (status.compareTo("ACTIVE") == 0) {
                tableStatus = true;
            }
            Thread.sleep(500);
        }

        List<ColumnDefinition> cols = response.schemaDefinition().allColumns();
        for (ColumnDefinition def : cols) {
            System.out.println("The column name is " + def.name());
            System.out.println("The column type is " + def.type());
        }

    } catch (KeyspacesException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```



- Per i dettagli sull'API, [GetTable](#) consulta AWS SDK for Java 2.x API Reference.

## ListKeyspaces

Il seguente esempio di codice mostra come utilizzare `ListKeyspaces`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public static void listKeyspacesPaginator(KeyspacesClient keyClient) {
    try {
        ListKeyspacesRequest keyspacesRequest = ListKeyspacesRequest.builder()
            .maxResults(10)
            .build();

        ListKeyspacesIterable listRes =
keyClient.listKeyspacesPaginator(keyspacesRequest);
        listRes.stream()
            .flatMap(r -> r.keyspaces().stream())
            .forEach(content -> System.out.println(" Name: " +
content.keyspaceName()));

    } catch (KeyspacesException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Per i dettagli sull'API, [ListKeyspaces](#) consulta AWS SDK for Java 2.x API Reference.

## ListTables

Il seguente esempio di codice mostra come utilizzare `ListTables`.

## SDK per Java 2.x

 Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public static void listTables(KeyspacesClient keyClient, String keyspaceName) {
    try {
        ListTablesRequest tablesRequest = ListTablesRequest.builder()
            .keyspaceName(keyspaceName)
            .build();

        ListTablesIterable listRes =
keyClient.listTablesPaginator(tablesRequest);
        listRes.stream()
            .flatMap(r -> r.tables().stream())
            .forEach(content -> System.out.println(" ARN: " +
content.resourceArn() +
                " Table name: " + content.tableName()));

    } catch (KeyspacesException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Per i dettagli sull'API, [ListTables](#) consulta AWS SDK for Java 2.x API Reference.

## RestoreTable

Il seguente esempio di codice mostra come utilizzare `RestoreTable`.

## SDK per Java 2.x

 Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public static void restoreTable(KeyspacesClient keyClient, String keyspaceName,
    ZonedDateTime utc) {
    try {
        Instant myTime = utc.toInstant();
        RestoreTableRequest restoreTableRequest = RestoreTableRequest.builder()
            .restoreTimestamp(myTime)
            .sourceTableName("Movie")
            .targetKeyspaceName(keyspaceName)
            .targetTableName("MovieRestore")
            .sourceKeyspaceName(keyspaceName)
            .build();

        RestoreTableResponse response =
            keyClient.restoreTable(restoreTableRequest);
        System.out.println("The ARN of the restored table is " +
            response.restoredTableARN());

    } catch (KeyspacesException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Per i dettagli sull'API, [RestoreTable](#) consulta AWS SDK for Java 2.x API Reference.

## UpdateTable

Il seguente esempio di codice mostra come utilizzare `UpdateTable`.

## SDK per Java 2.x

 Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public static void updateTable(KeyspacesClient keyClient, String keySpace,
String tableName) {
    try {
        ColumnDefinition def = ColumnDefinition.builder()
            .name("watched")
            .type("boolean")
            .build();

        UpdateTableRequest tableRequest = UpdateTableRequest.builder()
            .keyspaceName(keySpace)
            .tableName(tableName)
            .addColumnns(def)
            .build();

        keyClient.updateTable(tableRequest);

    } catch (KeyspacesException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Per i dettagli sull'API, [UpdateTable](#) consulta AWS SDK for Java 2.x API Reference.

## Scenari

Inizia con keyspace e tabelle

L'esempio di codice seguente mostra come:

- Crea uno spazio chiave e una tabella. Lo schema della tabella contiene i dati dei film e il point-in-time ripristino è abilitato.

- Connect al keyspace utilizzando una connessione TLS sicura con autenticazione SigV4.
- Esegui una query sulla tabella. Aggiungi, recupera e aggiorna i dati dei film.
- Aggiorna la tabella. Aggiungi una colonna per tenere traccia dei film guardati.
- Ripristina lo stato precedente della tabella e ripulisci le risorse.

## SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
/**
 * Before running this Java (v2) code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * Before running this Java code example, you must create a
 * Java keystore (JKS) file and place it in your project's resources folder.
 *
 * This file is a secure file format used to hold certificate information for
 * Java applications. This is required to make a connection to Amazon Keyspaces.
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/keyspaces/latest/devguide/using_java_driver.html
 *
 * This Java example performs the following tasks:
 *
 * 1. Create a keyspace.
 * 2. Check for keyspace existence.
 * 3. List keyspaces using a paginator.
 * 4. Create a table with a simple movie data schema and enable point-in-time
 * recovery.
 * 5. Check for the table to be in an Active state.
 * 6. List all tables in the keyspace.
 * 7. Use a Cassandra driver to insert some records into the Movie table.
```

- \* 8. Get all records from the Movie table.
- \* 9. Get a specific Movie.
- \* 10. Get a UTC timestamp for the current time.
- \* 11. Update the table schema to add a 'watched' Boolean column.
- \* 12. Update an item as watched.
- \* 13. Query for items with watched = True.
- \* 14. Restore the table back to the previous state using the timestamp.
- \* 15. Check for completion of the restore action.
- \* 16. Delete the table.
- \* 17. Confirm that both tables are deleted.
- \* 18. Delete the keyspace.
- \*/

```
public class ScenarioKeyspaces {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    /*
     * Usage:
     * fileName - The name of the JSON file that contains movie data. (Get this file
     * from the GitHub repo at resources/sample_file.)
     * keyspaceName - The name of the keyspace to create.
     */
    public static void main(String[] args) throws InterruptedException, IOException
    {
        String fileName = "<Replace with the JSON file that contains movie data>";
        String keyspaceName = "<Replace with the name of the keyspace to create>";
        String titleUpdate = "The Family";
        int yearUpdate = 2013;
        String tableName = "Movie";
        String tableNameRestore = "MovieRestore";
        Region region = Region.US_EAST_1;
        KeyspacesClient keyClient = KeyspacesClient.builder()
            .region(region)
            .build();

        DriverConfigLoader loader =
        DriverConfigLoader.fromClasspath("application.conf");
        CqlSession session = CqlSession.builder()
            .withConfigLoader(loader)
            .build();

        System.out.println(DASHES);
        System.out.println("Welcome to the Amazon Keyspaces example scenario.");
        System.out.println(DASHES);
    }
}
```

```
System.out.println(DASHES);
System.out.println("1. Create a keyspace.");
createKeySpace(keyClient, keyspaceName);
System.out.println(DASHES);

System.out.println(DASHES);
Thread.sleep(5000);
System.out.println("2. Check for keyspace existence.");
checkKeyspaceExistence(keyClient, keyspaceName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("3. List keyspace using a paginator.");
listKeyspacesPaginator(keyClient);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("4. Create a table with a simple movie data schema and
enable point-in-time recovery.");
createTable(keyClient, keyspaceName, tableName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("5. Check for the table to be in an Active state.");
Thread.sleep(6000);
checkTable(keyClient, keyspaceName, tableName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("6. List all tables in the keyspace.");
listTables(keyClient, keyspaceName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("7. Use a Cassandra driver to insert some records into
the Movie table.");
Thread.sleep(6000);
loadData(session, fileName, keyspaceName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("8. Get all records from the Movie table.");
getMovieData(session, keyspaceName);
```

```
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("9. Get a specific Movie.");
getSpecificMovie(session, keyspaceName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("10. Get a UTC timestamp for the current time.");
ZonedDateTime utc = ZonedDateTime.now(ZoneOffset.UTC);
System.out.println("DATETIME = " + Date.from(utc.toInstant()));
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("11. Update the table schema to add a watched Boolean
column.");
updateTable(keyClient, keyspaceName, tableName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("12. Update an item as watched.");
Thread.sleep(10000); // Wait 10 secs for the update.
updateRecord(session, keyspaceName, titleUpdate, yearUpdate);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("13. Query for items with watched = True.");
getWatchedData(session, keyspaceName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("14. Restore the table back to the previous state using
the timestamp.");
System.out.println("Note that the restore operation can take up to 20
minutes.");
restoreTable(keyClient, keyspaceName, utc);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("15. Check for completion of the restore action.");
Thread.sleep(5000);
checkRestoredTable(keyClient, keyspaceName, "MovieRestore");
System.out.println(DASHES);
```



```
        System.out.println(DASHES);
        System.out.println("16. Delete both tables.");
        deleteTable(keyClient, keyspaceName, tableName);
        deleteTable(keyClient, keyspaceName, tableNameRestore);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("17. Confirm that both tables are deleted.");
        checkTableDelete(keyClient, keyspaceName, tableName);
        checkTableDelete(keyClient, keyspaceName, tableNameRestore);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("18. Delete the keyspace.");
        deleteKeyspace(keyClient, keyspaceName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("The scenario has completed successfully.");
        System.out.println(DASHES);
    }

    public static void deleteKeyspace(KeyspacesClient keyClient, String
keyspaceName) {
        try {
            DeleteKeyspaceRequest deleteKeyspaceRequest =
DeleteKeyspaceRequest.builder()
                .keyspaceName(keyspaceName)
                .build();

            keyClient.deleteKeyspace(deleteKeyspaceRequest);

        } catch (KeyspacesException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void checkTableDelete(KeyspacesClient keyClient, String
keyspaceName, String tableName)
        throws InterruptedException {
        try {
            String status;
            GetTableResponse response;
```

```
        GetTableRequest tableRequest = GetTableRequest.builder()
            .keyspaceName(keyspaceName)
            .tableName(tableName)
            .build();

        // Keep looping until table cannot be found and a
ResourceNotFoundException is
        // thrown.
        while (true) {
            response = keyClient.getTable(tableRequest);
            status = response.statusAsString();
            System.out.println(". The table status is " + status);
            Thread.sleep(500);
        }

    } catch (ResourceNotFoundException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }
    System.out.println("The table is deleted");
}

public static void deleteTable(KeyspacesClient keyClient, String keyspaceName,
String tableName) {
    try {
        DeleteTableRequest tableRequest = DeleteTableRequest.builder()
            .keyspaceName(keyspaceName)
            .tableName(tableName)
            .build();

        keyClient.deleteTable(tableRequest);

    } catch (KeyspacesException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void checkRestoredTable(KeyspacesClient keyClient, String
keyspaceName, String tableName)
    throws InterruptedException {
    try {
        boolean tableStatus = false;
        String status;
        GetTableResponse response = null;
```

```
        GetTableRequest tableRequest = GetTableRequest.builder()
            .keyspaceName(keyspaceName)
            .tableName(tableName)
            .build();

        while (!tableStatus) {
            response = keyClient.getTable(tableRequest);
            status = response.statusAsString();
            System.out.println("The table status is " + status);

            if (status.compareTo("ACTIVE") == 0) {
                tableStatus = true;
            }
            Thread.sleep(500);
        }

        List<ColumnDefinition> cols = response.schemaDefinition().allColumns();
        for (ColumnDefinition def : cols) {
            System.out.println("The column name is " + def.name());
            System.out.println("The column type is " + def.type());
        }

    } catch (KeyspacesException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void restoreTable(KeyspacesClient keyClient, String keyspaceName,
    ZonedDateTime utc) {
    try {
        Instant myTime = utc.toInstant();
        RestoreTableRequest restoreTableRequest = RestoreTableRequest.builder()
            .restoreTimestamp(myTime)
            .sourceTableName("Movie")
            .targetKeyspaceName(keyspaceName)
            .targetTableName("MovieRestore")
            .sourceKeyspaceName(keyspaceName)
            .build();

        RestoreTableResponse response =
            keyClient.restoreTable(restoreTableRequest);
        System.out.println("The ARN of the restored table is " +
            response.restoredTableARN());
    }
}
```

```

    } catch (KeyspacesException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void getWatchedData(CqlSession session, String keyspaceName) {
    ResultSet resultSet = session
        .execute("SELECT * FROM \"" + keyspaceName + "\".\"Movie\" WHERE
watched = true ALLOW FILTERING;");
    resultSet.forEach(item -> {
        System.out.println("The Movie title is " + item.getString("title"));
        System.out.println("The Movie year is " + item.getInt("year"));
        System.out.println("The plot is " + item.getString("plot"));
    });
}

public static void updateRecord(CqlSession session, String keySpace, String
titleUpdate, int yearUpdate) {
    String sqlStatement = "UPDATE \"" + keySpace
        + "\".\"Movie\" SET watched=true WHERE title = :k0 AND year = :k1;";
    BatchStatementBuilder builder =
BatchStatement.builder(DefaultBatchType.UNLOGGED);
    builder.setConsistencyLevel(ConsistencyLevel.LOCAL_QUORUM);
    PreparedStatement preparedStatement = session.prepare(sqlStatement);
    builder.addStatement(preparedStatement.boundStatementBuilder()
        .setString("k0", titleUpdate)
        .setInt("k1", yearUpdate)
        .build());

    BatchStatement batchStatement = builder.build();
    session.execute(batchStatement);
}

public static void updateTable(KeyspacesClient keyClient, String keySpace,
String tableName) {
    try {
        ColumnDefinition def = ColumnDefinition.builder()
            .name("watched")
            .type("boolean")
            .build();

        UpdateTableRequest tableRequest = UpdateTableRequest.builder()

```

```

        .keyspaceName(keySpace)
        .tableName(tableName)
        .addColumn(def)
        .build();

        keyClient.updateTable(tableRequest);

    } catch (KeyspacesException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void getSpecificMovie(CqlSession session, String keyspaceName) {
    ResultSet resultSet = session.execute(
        "SELECT * FROM \"" + keyspaceName + "\".\"Movie\" WHERE title = 'The
Family' ALLOW FILTERING ;");
    resultSet.forEach(item -> {
        System.out.println("The Movie title is " + item.getString("title"));
        System.out.println("The Movie year is " + item.getInt("year"));
        System.out.println("The plot is " + item.getString("plot"));
    });
}

// Get records from the Movie table.
public static void getMovieData(CqlSession session, String keyspaceName) {
    ResultSet resultSet = session.execute("SELECT * FROM \"" + keyspaceName +
"\".\"Movie\";");
    resultSet.forEach(item -> {
        System.out.println("The Movie title is " + item.getString("title"));
        System.out.println("The Movie year is " + item.getInt("year"));
        System.out.println("The plot is " + item.getString("plot"));
    });
}

// Load data into the table.
public static void loadData(CqlSession session, String fileName, String
keySpace) throws IOException {
    String sqlStatement = "INSERT INTO \"" + keySpace + "\".\"Movie\" (title,
year, plot) values (:k0, :k1, :k2)";
    JsonParser parser = new JsonFactory().createParser(new File(fileName));
    com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
    Iterator<JsonNode> iter = rootNode.iterator();

```

```

    ObjectNode currentNode;
    int t = 0;
    while (iter.hasNext()) {

        // Add 20 movies to the table.
        if (t == 20)
            break;
        currentNode = (ObjectNode) iter.next();

        int year = currentNode.path("year").asInt();
        String title = currentNode.path("title").asText();
        String plot = currentNode.path("info").path("plot").toString();

        // Insert the data into the Amazon Keyspaces table.
        BatchStatementBuilder builder =
BatchStatement.builder(DefaultBatchType.UNLOGGED);
        builder.setConsistencyLevel(ConsistencyLevel.LOCAL_QUORUM);
        PreparedStatement preparedStatement = session.prepare(sqlStatement);
        builder.addStatement(preparedStatement.boundStatementBuilder()
            .setString("k0", title)
            .setInt("k1", year)
            .setString("k2", plot)
            .build());

        BatchStatement batchStatement = builder.build();
        session.execute(batchStatement);
        t++;
    }

    System.out.println("You have added " + t + " records successfully!");
}

public static void listTables(KeyspacesClient keyClient, String keyspaceName) {
    try {
        ListTablesRequest tablesRequest = ListTablesRequest.builder()
            .keyspaceName(keyspaceName)
            .build();

        ListTablesIterable listRes =
keyClient.listTablesPaginator(tablesRequest);
        listRes.stream()
            .flatMap(r -> r.tables().stream())
            .forEach(content -> System.out.println(" ARN: " +
content.resourceArn() +

```

```
        " Table name: " + content.tableName()));

    } catch (KeyspacesException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void checkTable(KeyspacesClient keyClient, String keyspaceName,
String tableName)
    throws InterruptedException {
    try {
        boolean tableStatus = false;
        String status;
        GetTableResponse response = null;
        GetTableRequest tableRequest = GetTableRequest.builder()
            .keyspaceName(keyspaceName)
            .tableName(tableName)
            .build();

        while (!tableStatus) {
            response = keyClient.getTable(tableRequest);
            status = response.statusAsString();
            System.out.println(". The table status is " + status);

            if (status.compareTo("ACTIVE") == 0) {
                tableStatus = true;
            }
            Thread.sleep(500);
        }

        List<ColumnDefinition> cols = response.schemaDefinition().allColumns();
        for (ColumnDefinition def : cols) {
            System.out.println("The column name is " + def.name());
            System.out.println("The column type is " + def.type());
        }

    } catch (KeyspacesException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

```
public static void createTable(KeyspacesClient keyClient, String keySpace,
String tableName) {
    try {
        // Set the columns.
        ColumnDefinition defTitle = ColumnDefinition.builder()
            .name("title")
            .type("text")
            .build();

        ColumnDefinition defYear = ColumnDefinition.builder()
            .name("year")
            .type("int")
            .build();

        ColumnDefinition defReleaseDate = ColumnDefinition.builder()
            .name("release_date")
            .type("timestamp")
            .build();

        ColumnDefinition defPlot = ColumnDefinition.builder()
            .name("plot")
            .type("text")
            .build();

        List<ColumnDefinition> collList = new ArrayList<>();
        collList.add(defTitle);
        collList.add(defYear);
        collList.add(defReleaseDate);
        collList.add(defPlot);

        // Set the keys.
        PartitionKey yearKey = PartitionKey.builder()
            .name("year")
            .build();

        PartitionKey titleKey = PartitionKey.builder()
            .name("title")
            .build();

        List<PartitionKey> keyList = new ArrayList<>();
        keyList.add(yearKey);
        keyList.add(titleKey);

        SchemaDefinition schemaDefinition = SchemaDefinition.builder()
```



```
        .partitionKeys(keyList)
        .allColumns(colList)
        .build();

    PointInTimeRecovery timeRecovery = PointInTimeRecovery.builder()
        .status(PointInTimeRecoveryStatus.ENABLED)
        .build();

    CreateTableRequest tableRequest = CreateTableRequest.builder()
        .keyspaceName(keySpace)
        .tableName(tableName)
        .schemaDefinition(schemaDefinition)
        .pointInTimeRecovery(timeRecovery)
        .build();

    CreateTableResponse response = keyClient.createTable(tableRequest);
    System.out.println("The table ARN is " + response.resourceArn());

} catch (KeyspacesException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}

public static void listKeyspacesPaginator(KeyspacesClient keyClient) {
    try {
        ListKeyspacesRequest keyspacesRequest = ListKeyspacesRequest.builder()
            .maxResults(10)
            .build();

        ListKeyspacesIterable listRes =
keyClient.listKeyspacesPaginator(keyspacesRequest);
        listRes.stream()
            .flatMap(r -> r.keyspaces().stream())
            .forEach(content -> System.out.println(" Name: " +
content.keyspaceName()));

    } catch (KeyspacesException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

```
public static void checkKeyspaceExistence(KeyspacesClient keyClient, String
keyspaceName) {
    try {
        GetKeyspaceRequest keyspaceRequest = GetKeyspaceRequest.builder()
            .keyspaceName(keyspaceName)
            .build();

        GetKeyspaceResponse response = keyClient.getKeyspace(keyspaceRequest);
        String name = response.keyspaceName();
        System.out.println("The " + name + " KeySpace is ready");

    } catch (KeyspacesException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void createKeySpace(KeyspacesClient keyClient, String
keyspaceName) {
    try {
        CreateKeyspaceRequest keyspaceRequest = CreateKeyspaceRequest.builder()
            .keyspaceName(keyspaceName)
            .build();

        CreateKeyspaceResponse response =
keyClient.createKeyspace(keyspaceRequest);
        System.out.println("The ARN of the KeySpace is " +
response.resourceArn());

    } catch (KeyspacesException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella Documentazione di riferimento delle API AWS SDK for Java 2.x .
  - [CreateKeyspace](#)
  - [CreateTable](#)
  - [DeleteKeyspace](#)

- [DeleteTable](#)
- [GetKeyspace](#)
- [GetTable](#)
- [ListKeyspaces](#)
- [ListTables](#)
- [RestoreTable](#)
- [UpdateTable](#)

## Esempi di Kinesis con SDK for Java 2.x

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando il AWS SDK for Java 2.x con Kinesis.

Le operazioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le operazioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati e negli esempi tra servizi.

Scenari: esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio.

Ogni esempio include un collegamento a GitHub, dove puoi trovare istruzioni su come configurare ed eseguire il codice nel contesto.

Argomenti

- [Azioni](#)
- [Esempi serverless](#)

Azioni

### **CreateStream**

Il seguente esempio di codice mostra come utilizzare `CreateStream`.

## SDK per Java 2.x

 Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.kinesis.KinesisClient;
import software.amazon.awssdk.services.kinesis.model.CreateStreamRequest;
import software.amazon.awssdk.services.kinesis.model.KinesisException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateDataStream {
    public static void main(String[] args) {

        final String usage = ""

                Usage:
                <streamName>

                Where:
                streamName - The Amazon Kinesis data stream (for example,
                StockTradeStream).
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String streamName = args[0];
        Region region = Region.US_EAST_1;
        KinesisClient kinesisClient = KinesisClient.builder()
```

```
        .region(region)
        .build();
    createStream(kinesisClient, streamName);
    System.out.println("Done");
    kinesisClient.close();
}

public static void createStream(KinesisClient kinesisClient, String streamName)
{
    try {
        CreateStreamRequest streamReq = CreateStreamRequest.builder()
            .streamName(streamName)
            .shardCount(1)
            .build();

        kinesisClient.createStream(streamReq);

    } catch (KinesisException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
}
```

- Per i dettagli sull'API, [CreateStream](#) consulta AWS SDK for Java 2.x API Reference.

## DeleteStream

Il seguente esempio di codice mostra come utilizzare `DeleteStream`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.kinesis.KinesisClient;
import software.amazon.awssdk.services.kinesis.model.DeleteStreamRequest;
```

```
import software.amazon.awssdk.services.kinesis.model.KinesisException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteDataStream {

    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <streamName>

            Where:
                streamName - The Amazon Kinesis data stream (for example,
StockTradeStream)
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String streamName = args[0];
        Region region = Region.US_EAST_1;
        KinesisClient kinesisClient = KinesisClient.builder()
            .region(region)
            .build();

        deleteStream(kinesisClient, streamName);
        kinesisClient.close();
        System.out.println("Done");
    }

    public static void deleteStream(KinesisClient kinesisClient, String streamName)
    {
        try {
            DeleteStreamRequest delStream = DeleteStreamRequest.builder()
                .streamName(streamName)

```

```

        .build();

        kinesisClient.deleteStream(delStream);

    } catch (KinesisException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
}
}

```

- Per i dettagli sull'API, [DeleteStream](#) consulta AWS SDK for Java 2.x API Reference.

## GetRecords

Il seguente esempio di codice mostra come utilizzare `GetRecords`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```

import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.kinesis.KinesisClient;
import software.amazon.awssdk.services.kinesis.model.DescribeStreamResponse;
import software.amazon.awssdk.services.kinesis.model.DescribeStreamRequest;
import software.amazon.awssdk.services.kinesis.model.Shard;
import software.amazon.awssdk.services.kinesis.model.GetShardIteratorRequest;
import software.amazon.awssdk.services.kinesis.model.GetShardIteratorResponse;
import software.amazon.awssdk.services.kinesis.model.Record;
import software.amazon.awssdk.services.kinesis.model.GetRecordsRequest;
import software.amazon.awssdk.services.kinesis.model.GetRecordsResponse;
import java.util.ArrayList;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development

```

```
* environment, including your credentials.
*
* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class GetRecords {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <streamName>

                Where:
                streamName - The Amazon Kinesis data stream to read from (for
example, StockTradeStream).
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String streamName = args[0];
        Region region = Region.US_EAST_1;
        KinesisClient kinesisClient = KinesisClient.builder()
                .region(region)
                .build();

        getStockTrades(kinesisClient, streamName);
        kinesisClient.close();
    }

    public static void getStockTrades(KinesisClient kinesisClient, String
streamName) {
        String shardIterator;
        String lastShardId = null;
        DescribeStreamRequest describeStreamRequest =
DescribeStreamRequest.builder()
                .streamName(streamName)
                .build();

        List<Shard> shards = new ArrayList<>();
        DescribeStreamResponse streamRes;
```



```
do {
    streamRes = kinesisClient.describeStream(describeStreamRequest);
    shards.addAll(streamRes.streamDescription().shards());

    if (shards.size() > 0) {
        lastShardId = shards.get(shards.size() - 1).shardId();
    }
} while (streamRes.streamDescription().hasMoreShards());

GetShardIteratorRequest itReq = GetShardIteratorRequest.builder()
    .streamName(streamName)
    .shardIteratorType("TRIM_HORIZON")
    .shardId(lastShardId)
    .build();

GetShardIteratorResponse shardIteratorResult =
kinesisClient.getShardIterator(itReq);
shardIterator = shardIteratorResult.shardIterator();

// Continuously read data records from shard.
List<Record> records;

// Create new GetRecordsRequest with existing shardIterator.
// Set maximum records to return to 1000.
GetRecordsRequest recordsRequest = GetRecordsRequest.builder()
    .shardIterator(shardIterator)
    .limit(1000)
    .build();

GetRecordsResponse result = kinesisClient.getRecords(recordsRequest);

// Put result into record list. Result may be empty.
records = result.records();

// Print records
for (Record record : records) {
    SdkBytes byteBuffer = record.data();
    System.out.printf("Seq No: %s - %s%n", record.sequenceNumber(), new
String(byteBuffer.asByteArray()));
}
}
```

- Per i dettagli sull'API, [GetRecords](#) consulta AWS SDK for Java 2.x API Reference.

## PutRecord

Il seguente esempio di codice mostra come utilizzare PutRecord.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.kinesis.KinesisClient;
import software.amazon.awssdk.services.kinesis.model.PutRecordRequest;
import software.amazon.awssdk.services.kinesis.model.KinesisException;
import software.amazon.awssdk.services.kinesis.model.DescribeStreamRequest;
import software.amazon.awssdk.services.kinesis.model.DescribeStreamResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class StockTradesWriter {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <streamName>

                Where:
                streamName - The Amazon Kinesis data stream to which records are
                written (for example, StockTradeStream)
                """;
    }
}
```

```
    if (args.length != 1) {
        System.out.println(usage);
        System.exit(1);
    }

    String streamName = args[0];
    Region region = Region.US_EAST_1;
    KinesisClient kinesisClient = KinesisClient.builder()
        .region(region)
        .build();

    // Ensure that the Kinesis Stream is valid.
    validateStream(kinesisClient, streamName);
    setStockData(kinesisClient, streamName);
    kinesisClient.close();
}

public static void setStockData(KinesisClient kinesisClient, String streamName)
{
    try {
        // Repeatedly send stock trades with a 100 milliseconds wait in between.
        StockTradeGenerator stockTradeGenerator = new StockTradeGenerator();

        // Put in 50 Records for this example.
        int index = 50;
        for (int x = 0; x < index; x++) {
            StockTrade trade = stockTradeGenerator.getRandomTrade();
            sendStockTrade(trade, kinesisClient, streamName);
            Thread.sleep(100);
        }

    } catch (KinesisException | InterruptedException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    System.out.println("Done");
}

private static void sendStockTrade(StockTrade trade, KinesisClient
kinesisClient,
    String streamName) {
    byte[] bytes = trade.toJsonAsBytes();
```

```

    // The bytes could be null if there is an issue with the JSON serialization
    by
    // the Jackson JSON library.
    if (bytes == null) {
        System.out.println("Could not get JSON bytes for stock trade");
        return;
    }

    System.out.println("Putting trade: " + trade);
    PutRecordRequest request = PutRecordRequest.builder()
        .partitionKey(trade.getTickerSymbol()) // We use the ticker symbol
as the partition key, explained in
                                                    // the Supplemental
Information section below.
        .streamName(streamName)
        .data(SdkBytes.fromByteArray(bytes))
        .build();

    try {
        kinesisClient.putRecord(request);
    } catch (KinesisException e) {
        System.err.println(e.getMessage());
    }
}

private static void validateStream(KinesisClient kinesisClient, String
streamName) {
    try {
        DescribeStreamRequest describeStreamRequest =
DescribeStreamRequest.builder()
            .streamName(streamName)
            .build();

        DescribeStreamResponse describeStreamResponse =
kinesisClient.describeStream(describeStreamRequest);

        if (!
describeStreamResponse.streamDescription().streamStatus().toString().equals("ACTIVE"))
        {
            System.err.println("Stream " + streamName + " is not active. Please
wait a few moments and try again.");
            System.exit(1);
        }
    }
}

```

```
        } catch (KinesisException e) {
            System.err.println("Error found while describing the stream " +
streamName);
            System.err.println(e);
            System.exit(1);
        }
    }
}
```

- Per i dettagli sull'API, [PutRecord](#) consulta AWS SDK for Java 2.x API Reference.

## Esempi serverless

### Richiamare una funzione Lambda da un trigger Kinesis

Il seguente esempio di codice mostra come implementare una funzione Lambda che riceve un evento attivato dalla ricezione di record da un flusso Kinesis. La funzione recupera il payload Kinesis, lo decodifica da Base64 e registra il contenuto del record.

### SDK per Java 2.x

#### Note

C'è altro su [GitHub](#) Trova l'esempio completo e scopri come eseguire la configurazione e l'esecuzione nel repository di [Esempi serverless](#).

### Utilizzo di un evento Kinesis con Lambda tramite Java.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package example;

import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.LambdaLogger;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.KinesisEvent;

public class Handler implements RequestHandler<KinesisEvent, Void> {
    @Override
    public Void handleRequest(final KinesisEvent event, final Context context) {
```

```
LambdaLogger logger = context.getLogger();
if (event.getRecords().isEmpty()) {
    logger.log("Empty Kinesis Event received");
    return null;
}
for (KinesisEvent.KinesisEventRecord record : event.getRecords()) {
    try {
        logger.log("Processed Event with EventId: "+record.getEventID());
        String data = new String(record.getKinesis().getData().array());
        logger.log("Data:"+ data);
        // TODO: Do interesting work based on the new data
    }
    catch (Exception ex) {
        logger.log("An error occurred:"+ex.getMessage());
        throw ex;
    }
}
logger.log("Successfully processed:"+event.getRecords().size()+" records");
return null;
}
}
```

## Segnalazione di errori di elementi batch per funzioni Lambda con un trigger Kinesis

Il seguente esempio di codice mostra come implementare una risposta batch parziale per le funzioni Lambda che ricevono eventi da un flusso Kinesis. La funzione riporta gli errori degli elementi batch nella risposta, segnalando a Lambda di riprovare tali messaggi in un secondo momento.

SDK per Java 2.x

### Note

C'è di più su [GitHub](#) Trova l'esempio completo e scopri come eseguire la configurazione e l'esecuzione nel repository di [Esempi serverless](#).

Segnalazione di errori di elementi batch di Kinesis con Lambda tramite Java.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
```

```
// SPDX-License-Identifier: Apache-2.0
import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.KinesisEvent;
import com.amazonaws.services.lambda.runtime.events.StreamsEventResponse;

import java.io.Serializable;
import java.util.ArrayList;
import java.util.List;

public class ProcessKinesisRecords implements RequestHandler<KinesisEvent,
StreamsEventResponse> {

    @Override
    public StreamsEventResponse handleRequest(KinesisEvent input, Context context) {

        List<StreamsEventResponse.BatchItemFailure> batchItemFailures = new
ArrayList<>();
        String curRecordSequenceNumber = "";

        for (KinesisEvent.KinesisEventRecord kinesisEventRecord :
input.getRecords()) {
            try {
                //Process your record
                KinesisEvent.Record kinesisRecord = kinesisEventRecord.getKinesis();
                curRecordSequenceNumber = kinesisRecord.getSequenceNumber();

            } catch (Exception e) {
                /* Since we are working with streams, we can return the failed item
immediately.
                Lambda will immediately begin to retry processing from this
failed item onwards. */
                batchItemFailures.add(new
StreamsEventResponse.BatchItemFailure(curRecordSequenceNumber));
                return new StreamsEventResponse(batchItemFailures);
            }
        }

        return new StreamsEventResponse(batchItemFailures);
    }
}
```

## AWS KMS esempi che utilizzano SDK for Java 2.x

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando AWS SDK for Java 2.x with AWS KMS.

Le operazioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le operazioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati e negli esempi tra servizi.

Scenari: esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio.

Ogni esempio include un collegamento a GitHub, dove è possibile trovare istruzioni su come configurare ed eseguire il codice nel contesto.

Nozioni di base

Chiave Hello KMS

Il seguente esempio di codice mostra come iniziare a utilizzare la chiave KMS.

SDK per Java 2.x

### Note

C'è altro su [GitHub](#) Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.kms.KmsClient;
import software.amazon.awssdk.services.kms.model.ListKeysRequest;
import software.amazon.awssdk.services.kms.model.KmsException;
import software.amazon.awssdk.services.kms.paginators.ListKeysIterable;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
```



```
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class HelloKMS {
    public static void main(String[] args) {
        Region region = Region.US_WEST_2;
        KmsClient kmsClient = KmsClient.builder()
            .region(region)
            .build();

        listAllKeys(kmsClient);
        kmsClient.close();
    }

    public static void listAllKeys(KmsClient kmsClient) {
        try {
            ListKeysRequest listKeysRequest = ListKeysRequest.builder()
                .limit(15)
                .build();

            ListKeysIterable keysResponse =
kmsClient.listKeysPaginator(listKeysRequest);
            keysResponse.stream()
                .flatMap(r -> r.keys().stream())
                .forEach(key -> System.out
                    .println(" The key ARN is: " + key.keyArn() + ". The key Id is:
" + key.keyId()));

        } catch (KmsException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Per i dettagli sull'API, [listKeysPaginator](#) consulta AWS SDK for Java 2.x API Reference.

## Argomenti

- [Azioni](#)
- [Scenari](#)

## Azioni

### CreateAlias

Il seguente esempio di codice mostra come utilizzare `CreateAlias`.

SDK per Java 2.x

#### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public static void createCustomAlias(KmsClient kmsClient, String targetKeyId,
String aliasName) {
    try {
        CreateAliasRequest aliasRequest = CreateAliasRequest.builder()
            .aliasName(aliasName)
            .targetKeyId(targetKeyId)
            .build();

        kmsClient.createAlias(aliasRequest);
        System.out.println(aliasName + " was successfully created.");

    } catch (ResourceExistsException e) {
        System.err.println("Alias already exists: " + e.getMessage());
        System.err.println("Moving on...");
    } catch (Exception e) {
        System.err.println("An unexpected error occurred: " + e.getMessage());
        System.err.println("Moving on...");
    }
}
```

- Per i dettagli sull'API, [CreateAlias](#) consulta AWS SDK for Java 2.x API Reference.

### CreateGrant

Il seguente esempio di codice mostra come utilizzare `CreateGrant`.

## SDK per Java 2.x

 Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public static String grantKey(KmsClient kmsClient, String keyId, String
granteePrincipal) {
    try {
        // Add the desired KMS Grant permissions.
        List<GrantOperation> grantPermissions = new ArrayList<>();
        grantPermissions.add(GrantOperation.ENCRYPT);
        grantPermissions.add(GrantOperation.DECRYPT);
        grantPermissions.add(GrantOperation.DESCRIBE_KEY);

        CreateGrantRequest grantRequest = CreateGrantRequest.builder()
            .keyId(keyId)
            .name("grant1")
            .granteePrincipal(granteePrincipal)
            .operations(grantPermissions)
            .build();

        CreateGrantResponse response = kmsClient.createGrant(grantRequest);
        return response.grantId();

    } catch (KmsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return "";
}
```

- Per i dettagli sull'API, [CreateGrant](#) consulta AWS SDK for Java 2.x API Reference.

## CreateKey

Il seguente esempio di codice mostra come utilizzare `CreateKey`.

## SDK per Java 2.x

 Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public static String createKey(KmsClient kmsClient, String keyDesc) {
    try {
        CreateKeyRequest keyRequest = CreateKeyRequest.builder()
            .description(keyDesc)
            .customerMasterKeySpec(CustomerMasterKeySpec.SYMMETRIC_DEFAULT)
            .keyUsage("ENCRYPT_DECRYPT")
            .build();

        CreateKeyResponse result = kmsClient.createKey(keyRequest);
        System.out.println("Symmetric key with ARN [" +
result.keyMetadata().arn() + "] has been created.");
        return result.keyMetadata().keyId();

    } catch (KmsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return "";
}
```

- Per i dettagli sull'API, [CreateKey](#) consulta AWS SDK for Java 2.x API Reference.

## Decrypt

Il seguente esempio di codice mostra come utilizzare `Decrypt`.

## SDK per Java 2.x

 Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public static String decryptData(KmsClient kmsClient, SdkBytes encryptedData,
String keyId) {
    try {
        DecryptRequest decryptRequest = DecryptRequest.builder()
            .ciphertextBlob(encryptedData)
            .keyId(keyId)
            .build();

        DecryptResponse decryptResponse = kmsClient.decrypt(decryptRequest);
        return decryptResponse.plaintext().asString(StandardCharsets.UTF_8);


    } catch (KmsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return "";
}
```

- Per i dettagli sull'API, [consulta Decrypt](#) in AWS SDK for Java 2.x API Reference.

## DeleteAlias

Il seguente esempio di codice mostra come utilizzare DeleteAlias

## SDK per Java 2.x

 Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public static void deleteSpecificAlias(KmsClient kmsClient, String aliasName) {
    try {
        DeleteAliasRequest deleteAliasRequest = DeleteAliasRequest.builder()
            .aliasName(aliasName)
            .build();

        kmsClient.deleteAlias(deleteAliasRequest);

    } catch (KmsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- Per i dettagli sull'API, [DeleteAlias](#) consulta AWS SDK for Java 2.x API Reference.

## DescribeKey

Il seguente esempio di codice mostra come utilizzare `DescribeKey`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public static boolean isKeyEnabled(KmsClient kmsClient, String keyId) {
    try {
        DescribeKeyRequest keyRequest = DescribeKeyRequest.builder()
            .keyId(keyId)
            .build();

        DescribeKeyResponse response = kmsClient.describeKey(keyRequest);
        KeyState keyState = response.keyMetadata().keyState();
        if (keyState == KeyState.ENABLED) {
            System.out.println("The key is enabled.");
            return true;
        } else {
```

```
        System.out.println("The key is not enabled. Key state: " +
keyState);
    }

    } catch (KmsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return false;
}
```

- Per i dettagli sull'API, [DescribeKey](#) consulta AWS SDK for Java 2.x API Reference.

## DisableKey

Il seguente esempio di codice mostra come utilizzare `DisableKey`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public static void disableKey(KmsClient kmsClient, String keyId) {
    try {
        DisableKeyRequest keyRequest = DisableKeyRequest.builder()
            .keyId(keyId)
            .build();

        kmsClient.disableKey(keyRequest);

    } catch (KmsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- Per i dettagli sull'API, [DisableKey](#) consulta AWS SDK for Java 2.x API Reference.

## EnableKey

Il seguente esempio di codice mostra come utilizzare `EnableKey`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
// Enable the KMS key.
public static void enableKey(KmsClient kmsClient, String keyId) {
    try {
        EnableKeyRequest enableKeyRequest = EnableKeyRequest.builder()
            .keyId(keyId)
            .build();

        kmsClient.enableKey(enableKeyRequest);

    } catch (KmsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- Per i dettagli sull'API, [EnableKey](#) consulta AWS SDK for Java 2.x API Reference.

## Encrypt

Il seguente esempio di codice mostra come utilizzare `Encrypt`.



## SDK per Java 2.x

 Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public static SdkBytes encryptData(KmsClient kmsClient, String keyId, String
text) {
    try {
        SdkBytes myBytes = SdkBytes.fromUtf8String(text);
        EncryptRequest encryptRequest = EncryptRequest.builder()
            .keyId(keyId)
            .plaintext(myBytes)
            .build();

        EncryptResponse response = kmsClient.encrypt(encryptRequest);
        String algorithm = response.encryptionAlgorithm().toString();
        System.out.println("The string was encrypted with algorithm " +
algorithm + ".");

        // Get the encrypted data.
        SdkBytes encryptedData = response.ciphertextBlob();
        return encryptedData;


    } catch (KmsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return null;
}
```

- Per i dettagli sull'API, consulta [Encrypt](#) in AWS SDK for Java 2.x API Reference.

## ListAliases

Il seguente esempio di codice mostra come utilizzare `ListAliases`.

## SDK per Java 2.x

 Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public static void listAllAliases(KmsClient kmsClient) {
    try {
        ListAliasesRequest aliasesRequest = ListAliasesRequest.builder()
            .limit(15)
            .build();

        ListAliasesIterable aliasesResponse =
kmsClient.listAliasesPaginator(aliasesRequest);
        aliasesResponse.stream()
            .flatMap(r -> r.aliases().stream())
            .forEach(alias -> System.out
                .println("The alias name is: " + alias.aliasName()));


    } catch (KmsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- Per i dettagli sull'API, [ListAliases](#) consulta AWS SDK for Java 2.x API Reference.

## ListGrants

Il seguente esempio di codice mostra come utilizzare `ListGrants`.

## SDK per Java 2.x

 Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public static void displayGrantIds(KmsClient kmsClient, String keyId) {
    try {
        ListGrantsRequest grantsRequest = ListGrantsRequest.builder()
            .keyId(keyId)
            .limit(15)
            .build();

        ListGrantsIterable response =
kmsClient.listGrantsPaginator(grantsRequest);
        response.stream()
            .flatMap(r -> r.grants().stream())
            .forEach(grant -> {
                System.out.println("The grant Id is : " + grant.grantId());
                List<GrantOperation> ops = grant.operations();
                for (GrantOperation op : ops) {
                    System.out.println(op.name());
                }
            });

    } catch (KmsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- Per i dettagli sull'API, [ListGrants](#) consulta AWS SDK for Java 2.x API Reference.

## ListKeyPolicies

Il seguente esempio di codice mostra come utilizzare `ListKeyPolicies`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public static void getKeyPolicy(KmsClient kmsClient, String keyId, String
policyName) {
    try {
        GetKeyPolicyRequest policyRequest = GetKeyPolicyRequest.builder()
            .keyId(keyId)
            .policyName(policyName)
            .build();

        GetKeyPolicyResponse response = kmsClient.getKeyPolicy(policyRequest);
        System.out.println("The response is "+response.policy());
    } catch (KmsException e) {
        if (e.getMessage().contains("No such policy exists")) {
            System.out.println("The policy cannot be found. Error message: " +
e.getMessage());
        } else {
            throw e;
        }
    }
}
```

- Per i dettagli sull'API, [ListKeyPolicies](#) consulta AWS SDK for Java 2.x API Reference.

## ListKeys

Il seguente esempio di codice mostra come utilizzare `ListKeys`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.kms.KmsClient;
import software.amazon.awssdk.services.kms.model.ListKeysRequest;
import software.amazon.awssdk.services.kms.model.KmsException;
import software.amazon.awssdk.services.kms.paginators.ListKeysIterable;
```

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class HelloKMS {
    public static void main(String[] args) {
        Region region = Region.US_WEST_2;
        KmsClient kmsClient = KmsClient.builder()
            .region(region)
            .build();

        listAllKeys(kmsClient);
        kmsClient.close();
    }

    public static void listAllKeys(KmsClient kmsClient) {
        try {
            ListKeysRequest listKeysRequest = ListKeysRequest.builder()
                .limit(15)
                .build();

            ListKeysIterable keysResponse =
kmsClient.listKeysPaginator(listKeysRequest);
            keysResponse.stream()
                .flatMap(r -> r.keys().stream())
                .forEach(key -> System.out
                    .println(" The key ARN is: " + key.keyArn() + ". The key Id is:
" + key.keyId()));

        } catch (KmsException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Per i dettagli sull'API, [ListKeys](#) consulta AWS SDK for Java 2.x API Reference.

## RevokeGrant

Il seguente esempio di codice mostra come utilizzare `RevokeGrant`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public static void revokeKeyGrant(KmsClient kmsClient, String keyId, String
grantId) {
    try {
        RevokeGrantRequest grantRequest = RevokeGrantRequest.builder()
            .keyId(keyId)
            .grantId(grantId)
            .build();

        kmsClient.revokeGrant(grantRequest);
        System.out.println("Grant ID: [" + grantId + "] was successfully
revoked!");
    } catch (KmsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- Per i dettagli sull'API, [RevokeGrant](#) consulta AWS SDK for Java 2.x API Reference.

## ScheduleKeyDeletion

Il seguente esempio di codice mostra come utilizzare `ScheduleKeyDeletion`.

## SDK per Java 2.x

 Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public static void deleteKey(KmsClient kmsClient, String keyId) {
    try {
        ScheduleKeyDeletionRequest deletionRequest =
ScheduleKeyDeletionRequest.builder()
        .keyId(keyId)
        .pendingWindowInDays(7)
        .build();

        kmsClient.scheduleKeyDeletion(deletionRequest);
        System.out.println("The key will be deleted in 7 days.");


    } catch (KmsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- Per i dettagli sull'API, [ScheduleKeyDeletion](#) consulta AWS SDK for Java 2.x API Reference.

## Sign

Il seguente esempio di codice mostra come utilizzare Sign.

## SDK per Java 2.x

 Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public static void signVerifyData(KmsClient kmsClient) {
    String signMessage = "Here is the message that will be digitally signed";

    // Create an AWS KMS key used to digitally sign data.
    CreateKeyRequest request = CreateKeyRequest.builder()
        .keySpec(KeySpec.RSA_2048) // Specify key spec
        .keyUsage(KeyUsageType.SIGN_VERIFY) // Specify key usage
        .origin(OriginType.AWS_KMS) // Specify key origin
        .build();

    CreateKeyResponse response = kmsClient.createKey(request);
    String keyId2 = response.keyMetadata().keyId();
    System.out.println("Created KMS key with ID: " + keyId2);

    SdkBytes bytes = SdkBytes.fromString(signMessage, Charset.defaultCharset());
    SignRequest signRequest = SignRequest.builder()
        .keyId(keyId2)
        .message(bytes)
        .signingAlgorithm(SigningAlgorithmSpec.RSASSA_PSS_SHA_256)
        .build();

    SignResponse signResponse = kmsClient.sign(signRequest);
    byte[] signedBytes = signResponse.signature().asByteArray();

    // Verify the digital signature.
    VerifyRequest verifyRequest = VerifyRequest.builder()
        .keyId(keyId2)

        .message(SdkBytes.fromByteArray(signMessage.getBytes(Charset.defaultCharset())))
        .signature(SdkBytes.fromByteBuffer(ByteBuffer.wrap(signedBytes)))
        .signingAlgorithm(SigningAlgorithmSpec.RSASSA_PSS_SHA_256)
        .build();

    VerifyResponse verifyResponse = kmsClient.verify(verifyRequest);
    System.out.println("Signature verification result: " +
verifyResponse.signatureValid());
}
```

- Per i dettagli sull'API, consulta [Sign](#) in AWS SDK for Java 2.x API Reference.



## TagResource

Il seguente esempio di codice mostra come usare `TagResource`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public static void tagKMSKey(KmsClient kmsClient, String keyId) {
    try {
        Tag tag = Tag.builder()
            .tagKey("Environment")
            .tagValue("Production")
            .build();

        TagResourceRequest tagResourceRequest = TagResourceRequest.builder()
            .keyId(keyId)
            .tags(tag)
            .build();

        kmsClient.tagResource(tagResourceRequest);
        System.out.println("The key has been tagged.");

    } catch (KmsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- Per i dettagli sull'API, [TagResource](#) consulta AWS SDK for Java 2.x API Reference.

## Scenari

Scopri le operazioni principali di KMS

L'esempio di codice seguente mostra come:

- Creare una chiave KMS.
- Elenca le chiavi KMS del tuo account e ottieni dettagli su di esse.
- Abilitare e disabilitare le chiavi KMS.
- Genera una chiave dati simmetrica che può essere utilizzata per la crittografia lato client.
- Genera una chiave asimmetrica utilizzata per firmare digitalmente i dati.
- Chiavi per tag.
- Eliminare le chiavi KMS.

## SDK per Java 2.x

### Note

C'è altro da fare. [GitHub](#) Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.kms.KmsClient;
import software.amazon.awssdk.services.kms.model.AliasListEntry;
import software.amazon.awssdk.services.kms.model.AlreadyExistsException;
import software.amazon.awssdk.services.kms.model.CreateAliasRequest;
import software.amazon.awssdk.services.kms.model.CreateGrantRequest;
import software.amazon.awssdk.services.kms.model.CreateGrantResponse;
import software.amazon.awssdk.services.kms.model.CreateKeyRequest;
import software.amazon.awssdk.services.kms.model.CreateKeyResponse;
import software.amazon.awssdk.services.kms.model.CustomerMasterKeySpec;
import software.amazon.awssdk.services.kms.model.DecryptRequest;
import software.amazon.awssdk.services.kms.model.DecryptResponse;
import software.amazon.awssdk.services.kms.model.DeleteAliasRequest;
import software.amazon.awssdk.services.kms.model.DescribeKeyRequest;
import software.amazon.awssdk.services.kms.model.DescribeKeyResponse;
import software.amazon.awssdk.services.kms.model.DisableKeyRequest;
import software.amazon.awssdk.services.kms.model.EnableKeyRequest;
import software.amazon.awssdk.services.kms.model.EnableKeyRotationRequest;
import software.amazon.awssdk.services.kms.model.EncryptRequest;
import software.amazon.awssdk.services.kms.model.EncryptResponse;
import software.amazon.awssdk.services.kms.model.GetKeyPolicyRequest;
import software.amazon.awssdk.services.kms.model.GetKeyPolicyResponse;
```

```
import software.amazon.awssdk.services.kms.model.GrantOperation;
import software.amazon.awssdk.services.kms.model.KeySpec;
import software.amazon.awssdk.services.kms.model.KeyState;
import software.amazon.awssdk.services.kms.model.KeyUsageType;
import software.amazon.awssdk.services.kms.model.KmsException;
import software.amazon.awssdk.services.kms.model.LimitExceededException;
import software.amazon.awssdk.services.kms.model.ListAliasesRequest;
import software.amazon.awssdk.services.kms.model.ListGrantsRequest;
import software.amazon.awssdk.services.kms.model.ListKeyPoliciesRequest;
import software.amazon.awssdk.services.kms.model.ListKeyPoliciesResponse;
import software.amazon.awssdk.services.kms.model.OriginType;
import software.amazon.awssdk.services.kms.model.PutKeyPolicyRequest;
import software.amazon.awssdk.services.kms.model.RevokeGrantRequest;
import software.amazon.awssdk.services.kms.model.ScheduleKeyDeletionRequest;
import software.amazon.awssdk.services.kms.model.SignRequest;
import software.amazon.awssdk.services.kms.model.SignResponse;
import software.amazon.awssdk.services.kms.model.SigningAlgorithmSpec;
import software.amazon.awssdk.services.kms.model.Tag;
import software.amazon.awssdk.services.kms.model.TagResourceRequest;
import software.amazon.awssdk.services.kms.model.VerifyRequest;
import software.amazon.awssdk.services.kms.model.VerifyResponse;
import software.amazon.awssdk.services.kms.paginators.ListAliasesIterable;
import software.amazon.awssdk.services.kms.paginators.ListGrantsIterable;
import software.amazon.awssdk.services.secretsmanager.model.ResourceExistsException;
import software.amazon.awssdk.services.sts.StsClient;
import software.amazon.awssdk.services.sts.model.GetCallerIdentityResponse;
import java.nio.ByteBuffer;
import java.nio.charset.Charset;
import java.nio.charset.StandardCharsets;
import java.util.List;
import java.util.ArrayList;
import java.util.Scanner;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class KMSScenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");
}
```

```
private static final String accountId = getAccountId();

public static void main(String[] args) {
    final String usage = ""
        Usage: <granteePrincipal>

        Where:
            granteePrincipal - The principal (user, service account, or
group) to whom the grant or permission is being given.
        """;

    if (args.length != 1) {
        System.out.println(usage);
        System.exit(1);
    }
    String granteePrincipal = args[0];
    String policyName = "default";

    Scanner scanner = new Scanner(System.in);
    String keyDesc = "Created by the AWS KMS API";

    Region region = Region.US_WEST_2;
    KmsClient kmsClient = KmsClient.builder()
        .region(region)
        .build();

    System.out.println(DASHES);
    System.out.println("")
        Welcome to the AWS Key Management SDK Getting Started scenario.

        This program demonstrates how to interact with AWS Key Management using
the AWS SDK for Java (v2).
        The AWS Key Management Service (KMS) is a secure and highly available
service that allows you to create
            and manage AWS KMS keys and control their use across a wide range of AWS
services and applications.
        KMS provides a centralized and unified approach to managing encryption
keys, making it easier to meet your
            data protection and regulatory compliance requirements.

        This Getting Started scenario creates two key types. A symmetric
encryption key is used to encrypt and decrypt data,
            and an asymmetric key used to digitally sign data.
        Let's get started...
```

```

        """);
        waitForInputToContinue(scanner);

        System.out.println(DASHES);
        System.out.println("1. Create a symmetric KMS key\n");
        System.out.println("First, the program will creates a symmetric KMS key that
you can used to encrypt and decrypt data.");
        waitForInputToContinue(scanner);
        String targetKeyId = createKey(kmsClient, keyDesc);
        waitForInputToContinue(scanner);

```

```

        System.out.println(DASHES);
        System.out.println("""
            2. Enable a KMS key

```

By default, when the SDK creates an AWS key it is enabled. The next bit of code checks to determine if the key is enabled. If it is not enabled, the code enables it.

```

        """);
        waitForInputToContinue(scanner);
        boolean isEnabled = isKeyEnabled(kmsClient, targetKeyId);
        if (!isEnabled)
            enableKey(kmsClient, targetKeyId);
        waitForInputToContinue(scanner);

        System.out.println(DASHES);
        System.out.println("3. Encrypt data using the symmetric KMS key");
        String plaintext = "Hello, AWS KMS!";
        System.out.printf("""

```

One of the main uses of symmetric keys is to encrypt and decrypt data.

Next, the code encrypts the string '%s' with the SYMMETRIC\_DEFAULT encryption algorithm.

```

            %n""", plaintext);
        waitForInputToContinue(scanner);
        SdkBytes ciphertext = encryptData(kmsClient, targetKeyId, plaintext);
        waitForInputToContinue(scanner);

        System.out.println(DASHES);
        System.out.println("4. Create an alias");
        System.out.println("""

```

```

        Enter an alias name for the key. The name should be prefixed with
        'alias/'.
        For example, 'alias/myFirstKey'.
        """);

        String aliasName = scanner.nextLine();
        String fullAliasName = aliasName.isEmpty() ? "alias/dev-encryption-key" :
aliasName;
        createCustomAlias(kmsClient, targetKeyId, fullAliasName);
        waitForInputToContinue(scanner);

        System.out.println(DASHES);
        System.out.println("5. List all of your aliases");
        waitForInputToContinue(scanner);
        listAllAliases(kmsClient);
        waitForInputToContinue(scanner);

        System.out.println(DASHES);
        System.out.println("6. Enable automatic rotation of the KMS key");
        System.out.println("")

```

By default, when the SDK enables automatic rotation of a KMS key, KMS rotates the key material of the KMS key one year (approximately 365 days) from the enable date and every year thereafter.

```

        """);
        waitForInputToContinue(scanner);
        enableKeyRotation(kmsClient, targetKeyId);
        waitForInputToContinue(scanner);

```

```

        System.out.println(DASHES);
        System.out.println("")
        7. Create a grant

```

A grant is a policy instrument that allows Amazon Web Services principals to use KMS keys.

It also can allow them to view a KMS key (DescribeKey) and create and manage grants.

When authorizing access to a KMS key, grants are considered along with key policies and IAM policies.

```

        """);

        waitForInputToContinue(scanner);
        String grantId = grantKey(kmsClient, targetKeyId, granteePrincipal);

```

```

    System.out.println("The code granted principal with ARN [" +
granteePrincipal + "] ");
    System.out.println("use of the symmetric key. The grant ID is [" + grantId +
"]");
    waitForInputToContinue(scanner);

    System.out.println(DASHES);
    System.out.println("8. List grants for the KMS key");
    waitForInputToContinue(scanner);
    displayGrantIds(kmsClient, targetKeyId);
    waitForInputToContinue(scanner);

    System.out.println(DASHES);
    System.out.println("9. Revoke the grant");
    waitForInputToContinue(scanner);
    revokeKeyGrant(kmsClient, targetKeyId, grantId);
    waitForInputToContinue(scanner);

    System.out.println(DASHES);
    System.out.println("10. Decrypt the data\n");
    System.out.println("""
        Lets decrypt the data that was encrypted in an early step.
        The code uses the same key to decrypt the string that we encrypted
earlier in the program.
        """);
    waitForInputToContinue(scanner);
    String decryptText = decryptData(kmsClient, ciphertext, targetKeyId);
    System.out.println("Decrypted text is: " + decryptText);
    waitForInputToContinue(scanner);

    System.out.println(DASHES);
    System.out.println("11. Replace a key policy\n");
    System.out.println("""
        A key policy is a resource policy for a KMS key. Key policies are the
primary way to control
        access to KMS keys. Every KMS key must have exactly one key policy. The
statements in the key policy
        determine who has permission to use the KMS key and how they can use
it.
        You can also use IAM policies and grants to control access to the KMS
key, but every KMS key
        must have a key policy.
    """);

```

By default, when you create a key by using the SDK, a policy is created that gives the AWS account that owns the KMS key full access to the KMS key. Let's try to replace the automatically created policy with the following policy.

```

        "Version": "2012-10-17",
        "Statement": [{
            "Effect": "Allow",
            "Principal": {"AWS": "arn:aws:iam::000000000000:root"},
            "Action": "kms:*",
            "Resource": "*"
        }]
    """);

    waitForInputToContinue(scanner);
    boolean polAdded = replacePolicy(kmsClient, targetKeyId, policyName);
    waitForInputToContinue(scanner);

    System.out.println(DASHES);
    System.out.println("12. Get the key policy\n");
    System.out.println("The next bit of code that runs gets the key policy to
make sure it exists.");
    waitForInputToContinue(scanner);
    getKeyPolicy(kmsClient, targetKeyId, policyName);
    waitForInputToContinue(scanner);

    System.out.println(DASHES);
    System.out.println("13. Create an asymmetric KMS key and sign your data\n");
    System.out.println("""
        Signing your data with an AWS key can provide several benefits that make
it an attractive option
        for your data signing needs. By using an AWS KMS key, you can leverage
the
        security controls and compliance features provided by AWS,
        which can help you meet various regulatory requirements and enhance the
overall security posture
        of your organization.
    """);
    waitForInputToContinue(scanner);
    signVerifyData(kmsClient);
    waitForInputToContinue(scanner);

```



```
System.out.println(DASHES);
System.out.println("14. Tag your symmetric KMS Key\n");
System.out.println("""
    By using tags, you can improve the overall management, security, and
governance of your
    KMS keys, making it easier to organize, track, and control access to
your encrypted data within
    your AWS environment
    """);
waitForInputToContinue(scanner);
tagKMSKey(kmsClient, targetKeyId);
waitForInputToContinue(scanner);

System.out.println(DASHES);
System.out.println("15. Schedule the deletion of the KMS key\n");
System.out.println("""
    By default, KMS applies a waiting period of 30 days,
    but you can specify a waiting period of 7-30 days. When this operation
is successful,
    the key state of the KMS key changes to PendingDeletion and the key
can't be used in any
    cryptographic operations. It remains in this state for the duration of
the waiting period.

    Deleting a KMS key is a destructive and potentially dangerous operation.
When a KMS key is deleted,
    all data that was encrypted under the KMS key is unrecoverable.\s
    """);
System.out.println("Would you like to delete the Key Management resources?
(y/n)");
String delAns = scanner.nextLine().trim();
if (delAns.equalsIgnoreCase("y")) {
    System.out.println("You selected to delete the AWS KMS resources.");
    waitForInputToContinue(scanner);
    deleteSpecificAlias(kmsClient, fullAliasName);
    disableKey(kmsClient, targetKeyId);
    deleteKey(kmsClient, targetKeyId);
} else {
    System.out.println("The Key Management resources will not be deleted");
}

System.out.println(DASHES);
```

```
        System.out.println("This concludes the AWS Key Management SDK Getting
Started scenario");
        System.out.println(DASHES);
    }
    public static void listAllAliases(KmsClient kmsClient) {
        try {
            ListAliasesRequest aliasesRequest = ListAliasesRequest.builder()
                .limit(15)
                .build();

            ListAliasesIterable aliasesResponse =
kmsClient.listAliasesPaginator(aliasesRequest);
            aliasesResponse.stream()
                .flatMap(r -> r.aliases().stream())
                .forEach(alias -> System.out
                    .println("The alias name is: " + alias.aliasName()));

        } catch (KmsException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void disableKey(KmsClient kmsClient, String keyId) {
        try {
            DisableKeyRequest keyRequest = DisableKeyRequest.builder()
                .keyId(keyId)
                .build();

            kmsClient.disableKey(keyRequest);

        } catch (KmsException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void signVerifyData(KmsClient kmsClient) {
        String signMessage = "Here is the message that will be digitally signed";

        // Create an AWS KMS key used to digitally sign data.
        CreateKeyRequest request = CreateKeyRequest.builder()
            .keySpec(KeySpec.RSA_2048) // Specify key spec
            .keyUsage(KeyUsageType.SIGN_VERIFY) // Specify key usage
```

```
        .origin(OriginType.AWS_KMS) // Specify key origin
        .build();

CreateKeyResponse response = kmsClient.createKey(request);
String keyId2 = response.keyMetadata().keyId();
System.out.println("Created KMS key with ID: " + keyId2);

SdkBytes bytes = SdkBytes.fromString(signMessage, Charset.defaultCharset());
SignRequest signRequest = SignRequest.builder()
    .keyId(keyId2)
    .message(bytes)
    .signingAlgorithm(SigningAlgorithmSpec.RSASSA_PSS_SHA_256)
    .build();

SignResponse signResponse = kmsClient.sign(signRequest);
byte[] signedBytes = signResponse.signature().asByteArray();

// Verify the digital signature.
VerifyRequest verifyRequest = VerifyRequest.builder()
    .keyId(keyId2)

    .message(SdkBytes.fromByteArray(signMessage.getBytes(Charset.defaultCharset())))
    .signature(SdkBytes.fromByteBuffer(ByteBuffer.wrap(signedBytes)))
    .signingAlgorithm(SigningAlgorithmSpec.RSASSA_PSS_SHA_256)
    .build();

VerifyResponse verifyResponse = kmsClient.verify(verifyRequest);
System.out.println("Signature verification result: " +
verifyResponse.signatureValid());
}

public static void tagKMSKey(KmsClient kmsClient, String keyId) {
    try {
        Tag tag = Tag.builder()
            .tagKey("Environment")
            .tagValue("Production")
            .build();

        TagResourceRequest tagResourceRequest = TagResourceRequest.builder()
            .keyId(keyId)
            .tags(tag)
            .build();

        kmsClient.tagResource(tagResourceRequest);
    }
}
```

```
        System.out.println("The key has been tagged.");

    } catch (KmsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void getKeyPolicy(KmsClient kmsClient, String keyId, String
policyName) {
    try {
        GetKeyPolicyRequest policyRequest = GetKeyPolicyRequest.builder()
            .keyId(keyId)
            .policyName(policyName)
            .build();

        GetKeyPolicyResponse response = kmsClient.getKeyPolicy(policyRequest);
        System.out.println("The response is "+response.policy());
    } catch (KmsException e) {
        if (e.getMessage().contains("No such policy exists")) {
            System.out.println("The policy cannot be found. Error message: " +
e.getMessage());
        } else {
            throw e;
        }
    }
}

public static boolean replacePolicy(KmsClient kmsClient, String keyId, String
policyName) {
    // Change the principle in the below JSON.
    String policy = ""
    {
        "Version": "2012-10-17",
        "Statement": [{
            "Effect": "Allow",
            "Principal": {"AWS": "arn:aws:iam::%s:root"},
            "Action": "kms:*",
            "Resource": "*"
        }
    ]
    }
    """".formatted(accountId);

    try {
```

```
        PutKeyPolicyRequest keyPolicyRequest = PutKeyPolicyRequest.builder()
            .keyId(keyId)
            .policyName(policyName)
            .policy(policy)
            .build();
        kmsClient.putKeyPolicy(keyPolicyRequest);
        System.out.println("The key policy has been replaced.");
    } catch (LimitExceededException e) {
        System.out.println("Policy limit reached. Unable to create the
policy.");
        return false;
    } catch (AlreadyExistsException e) {
        System.out.println("Only one policy per key is supported. Unable to
create the policy.");
        return false;
    } catch (KmsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }

    return true;
}

public static boolean doesKeyHavePolicy(KmsClient kmsClient, String keyId,
String policyName){
    ListKeyPoliciesRequest policiesRequest = ListKeyPoliciesRequest.builder()
        .keyId(keyId)
        .build();

    boolean hasPolicy = false;
    ListKeyPoliciesResponse response =
kmsClient.listKeyPolicies(policiesRequest);
    List<String>policyNames = response.policyNames();
    for (String pol : policyNames) {
        hasPolicy = true;
    }
    return hasPolicy;
}

public static void deleteKey(KmsClient kmsClient, String keyId) {
    try {
        ScheduleKeyDeletionRequest deletionRequest =
ScheduleKeyDeletionRequest.builder()
            .keyId(keyId)
```

```
        .pendingWindowInDays(7)
        .build();

        kmsClient.scheduleKeyDeletion(deletionRequest);
        System.out.println("The key will be deleted in 7 days.");

    } catch (KmsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void deleteSpecificAlias(KmsClient kmsClient, String aliasName) {
    try {
        DeleteAliasRequest deleteAliasRequest = DeleteAliasRequest.builder()
            .aliasName(aliasName)
            .build();

        kmsClient.deleteAlias(deleteAliasRequest);

    } catch (KmsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static boolean isKeyEnabled(KmsClient kmsClient, String keyId) {
    try {
        DescribeKeyRequest keyRequest = DescribeKeyRequest.builder()
            .keyId(keyId)
            .build();

        DescribeKeyResponse response = kmsClient.describeKey(keyRequest);
        KeyState keyState = response.keyMetadata().keyState();
        if (keyState == KeyState.ENABLED) {
            System.out.println("The key is enabled.");
            return true;
        } else {
            System.out.println("The key is not enabled. Key state: " +
keyState);
        }

    } catch (KmsException e) {
        System.err.println(e.getMessage());
    }
}
```

```
        System.exit(1);
    }
    return false;
}

public static String decryptData(KmsClient kmsClient, SdkBytes encryptedData,
String keyId) {
    try {
        DecryptRequest decryptRequest = DecryptRequest.builder()
            .ciphertextBlob(encryptedData)
            .keyId(keyId)
            .build();

        DecryptResponse decryptResponse = kmsClient.decrypt(decryptRequest);
        return decryptResponse.plaintext().asString(StandardCharsets.UTF_8);

    } catch (KmsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return "";
}

public static void revokeKeyGrant(KmsClient kmsClient, String keyId, String
grantId) {
    try {
        RevokeGrantRequest grantRequest = RevokeGrantRequest.builder()
            .keyId(keyId)
            .grantId(grantId)
            .build();

        kmsClient.revokeGrant(grantRequest);
        System.out.println("Grant ID: [" + grantId + "] was successfully
revoke!");

    } catch (KmsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void displayGrantIds(KmsClient kmsClient, String keyId) {
    try {
        ListGrantsRequest grantsRequest = ListGrantsRequest.builder()
```

```

        .keyId(keyId)
        .limit(15)
        .build();

    ListGrantsIterable response =
kmsClient.listGrantsPaginator(grantsRequest);
    response.stream()
        .flatMap(r -> r.grants().stream())
        .forEach(grant -> {
            System.out.println("The grant Id is : " + grant.grantId());
            List<GrantOperation> ops = grant.operations();
            for (GrantOperation op : ops) {
                System.out.println(op.name());
            }
        });

    } catch (KmsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static String grantKey(KmsClient kmsClient, String keyId, String
granteePrincipal) {
    try {
        // Add the desired KMS Grant permissions.
        List<GrantOperation> grantPermissions = new ArrayList<>();
        grantPermissions.add(GrantOperation.ENCRYPT);
        grantPermissions.add(GrantOperation.DECRYPT);
        grantPermissions.add(GrantOperation.DESCRIBE_KEY);

        CreateGrantRequest grantRequest = CreateGrantRequest.builder()
            .keyId(keyId)
            .name("grant1")
            .granteePrincipal(granteePrincipal)
            .operations(grantPermissions)
            .build();

        CreateGrantResponse response = kmsClient.createGrant(grantRequest);
        return response.grantId();

    } catch (KmsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

```



```
    }
    return "";
}

public static void enableKeyRotation(KmsClient kmsClient, String keyId) {
    try {
        EnableKeyRotationRequest enableKeyRotationRequest =
        EnableKeyRotationRequest.builder()
            .keyId(keyId)
            .build();

        kmsClient.enableKeyRotation(enableKeyRotationRequest);
        System.out.println("Key rotation has been enabled for key with id [" +
        keyId + "]");

    } catch (KmsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void createCustomAlias(KmsClient kmsClient, String targetKeyId,
String aliasName) {
    try {
        CreateAliasRequest aliasRequest = CreateAliasRequest.builder()
            .aliasName(aliasName)
            .targetKeyId(targetKeyId)
            .build();

        kmsClient.createAlias(aliasRequest);
        System.out.println(aliasName + " was successfully created.");

    } catch (ResourceExistsException e) {
        System.err.println("Alias already exists: " + e.getMessage());
        System.err.println("Moving on...");
    } catch (Exception e) {
        System.err.println("An unexpected error occurred: " + e.getMessage());
        System.err.println("Moving on...");
    }
}

public static SdkBytes encryptData(KmsClient kmsClient, String keyId, String
text) {
    try {
```

```
    SdkBytes myBytes = SdkBytes.fromUtf8String(text);
    EncryptRequest encryptRequest = EncryptRequest.builder()
        .keyId(keyId)
        .plaintext(myBytes)
        .build();

    EncryptResponse response = kmsClient.encrypt(encryptRequest);
    String algorithm = response.encryptionAlgorithm().toString();
    System.out.println("The string was encrypted with algorithm " +
algorithm + ".");

    // Get the encrypted data.
    SdkBytes encryptedData = response.ciphertextBlob();
    return encryptedData;

} catch (KmsException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
return null;
}

public static String createKey(KmsClient kmsClient, String keyDesc) {
    try {
        CreateKeyRequest keyRequest = CreateKeyRequest.builder()
            .description(keyDesc)
            .customerMasterKeySpec(CustomerMasterKeySpec.SYMMETRIC_DEFAULT)
            .keyUsage("ENCRYPT_DECRYPT")
            .build();

        CreateKeyResponse result = kmsClient.createKey(keyRequest);
        System.out.println("Symmetric key with ARN [" +
result.keyMetadata().arn() + "] has been created.");
        return result.keyMetadata().keyId();

    } catch (KmsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return "";
}

// Enable the KMS key.
public static void enableKey(KmsClient kmsClient, String keyId) {
```

```
    try {
        EnableKeyRequest enableKeyRequest = EnableKeyRequest.builder()
            .keyId(keyId)
            .build();

        kmsClient.enableKey(enableKeyRequest);

    } catch (KmsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

private static void waitForInputToContinue(Scanner scanner) {
    while (true) {
        System.out.println("");
        System.out.println("Enter 'c' followed by <ENTER> to continue:");
        String input = scanner.nextLine();

        if (input.trim().equalsIgnoreCase("c")) {
            System.out.println("Continuing with the program...");
            System.out.println("");
            break;
        } else {
            // Handle invalid input.
            System.out.println("Invalid input. Please try again.");
        }
    }
}

private static String getAccountId(){
    try (StsClient stsClient = StsClient.create()){
        GetCallerIdentityResponse callerIdentity =
stsClient.getCallerIdentity();
        return callerIdentity.account();
    }
}
}
```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella Documentazione di riferimento delle API AWS SDK for Java 2.x .
  - [CreateKey](#)
  - [DescribeKey](#)

- [DisableKey](#)
- [EnableKey](#)
- [GenerateDataKey](#)
- [ListKeys](#)
- [ScheduleKeyDeletion](#)
- [Sign](#)

## Esempi di Lambda con SDK for Java 2.x

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando AWS SDK for Java 2.x with Lambda.

Le operazioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le operazioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati e negli esempi tra servizi.

Scenari: esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio.

Ogni esempio include un collegamento a GitHub, dove è possibile trovare istruzioni su come configurare ed eseguire il codice nel contesto.

Nozioni di base

Hello Lambda

L'esempio di codice seguente mostra come iniziare a utilizzare Lambda.

SDK per Java 2.x

### Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
package com.example.lambda;
```

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.lambda.LambdaClient;
import software.amazon.awssdk.services.lambda.model.LambdaException;
import software.amazon.awssdk.services.lambda.model.ListFunctionsResponse;
import software.amazon.awssdk.services.lambda.model.FunctionConfiguration;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListLambdaFunctions {
    public static void main(String[] args) {
        Region region = Region.US_WEST_2;
        LambdaClient awsLambda = LambdaClient.builder()
            .region(region)
            .build();

        listFunctions(awsLambda);
        awsLambda.close();
    }

    public static void listFunctions(LambdaClient awsLambda) {
        try {
            ListFunctionsResponse functionResult = awsLambda.listFunctions();
            List<FunctionConfiguration> list = functionResult.functions();
            for (FunctionConfiguration config : list) {
                System.out.println("The function name is " + config.functionName());
            }
        } catch (LambdaException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Per i dettagli sull'API, [ListFunctions](#) consulta AWS SDK for Java 2.x API Reference.

## Argomenti

- [Azioni](#)
- [Scenari](#)
- [Esempi serverless](#)

## Azioni

### CreateFunction

Il seguente esempio di codice mostra come utilizzare `CreateFunction`.

SDK per Java 2.x

#### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.lambda.LambdaClient;
import software.amazon.awssdk.services.lambda.model.CreateFunctionRequest;
import software.amazon.awssdk.services.lambda.model.FunctionCode;
import software.amazon.awssdk.services.lambda.model.CreateFunctionResponse;
import software.amazon.awssdk.services.lambda.model.GetFunctionRequest;
import software.amazon.awssdk.services.lambda.model.GetFunctionResponse;
import software.amazon.awssdk.services.lambda.model.LambdaException;
import software.amazon.awssdk.services.lambda.model.Runtime;
import software.amazon.awssdk.services.lambda.waiters.LambdaWaiter;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;

/**
 * This code example requires a ZIP or JAR that represents the code of the
 * Lambda function.
 * If you do not have a ZIP or JAR, please refer to the following document:
 *

```

```
* https://github.com/aws-doc-sdk-examples/tree/master/javav2/usecases/creating\_workflows\_stepfunctions
*
* Also, set up your development environment, including your credentials.
*
* For information, see this documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/

public class CreateFunction {
    public static void main(String[] args) {

        final String usage = ""

            Usage:
                <functionName> <filePath> <role> <handler>\s

            Where:
                functionName - The name of the Lambda function.\s
                filePath - The path to the ZIP or JAR where the code is located.
\s
                role - The role ARN that has Lambda permissions.\s
                handler - The fully qualified method name (for example,
example.Handler::handleRequest). \s
                """;

        if (args.length != 4) {
            System.out.println(usage);
            System.exit(1);
        }

        String functionName = args[0];
        String filePath = args[1];
        String role = args[2];
        String handler = args[3];
        Region region = Region.US_WEST_2;
        LambdaClient awsLambda = LambdaClient.builder()
            .region(region)
            .build();

        createLambdaFunction(awsLambda, functionName, filePath, role, handler);
        awsLambda.close();
    }
}
```

```
public static void createLambdaFunction(LambdaClient awsLambda,
    String functionName,
    String filePath,
    String role,
    String handler) {

    try {
        LambdaWaiter waiter = awsLambda.waiter();
        InputStream is = new FileInputStream(filePath);
        SdkBytes fileToUpload = SdkBytes.fromInputStream(is);

        FunctionCode code = FunctionCode.builder()
            .zipFile(fileToUpload)
            .build();

        CreateFunctionRequest functionRequest = CreateFunctionRequest.builder()
            .functionName(functionName)
            .description("Created by the Lambda Java API")
            .code(code)
            .handler(handler)
            .runtime(Runtime.JAVA8)
            .role(role)
            .build();

        // Create a Lambda function using a waiter.
        CreateFunctionResponse functionResponse =
awsLambda.createFunction(functionRequest);
        GetFunctionRequest getFunctionRequest = GetFunctionRequest.builder()
            .functionName(functionName)
            .build();

        WaiterResponse<GetFunctionResponse> waiterResponse =
waiter.waitUntilFunctionExists(getFunctionRequest);
        waiterResponse.matched().response().ifPresent(System.out::println);
        System.out.println("The function ARN is " +
functionResponse.functionArn());

    } catch (LambdaException | FileNotFoundException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
}
```



- Per i dettagli sull'API, [CreateFunction](#) consulta AWS SDK for Java 2.x API Reference.

## DeleteFunction

Il seguente esempio di codice mostra come utilizzare `DeleteFunction`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.services.lambda.LambdaClient;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.lambda.model.DeleteFunctionRequest;
import software.amazon.awssdk.services.lambda.model.LambdaException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteFunction {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <functionName>\s

                Where:
                functionName - The name of the Lambda function.\s
                """;

        if (args.length != 1) {
```

```
        System.out.println(usage);
        System.exit(1);
    }

    String functionName = args[0];
    Region region = Region.US_EAST_1;
    LambdaClient awsLambda = LambdaClient.builder()
        .region(region)
        .build();

    deleteLambdaFunction(awsLambda, functionName);
    awsLambda.close();
}

public static void deleteLambdaFunction(LambdaClient awsLambda, String
functionName) {
    try {
        DeleteFunctionRequest request = DeleteFunctionRequest.builder()
            .functionName(functionName)
            .build();

        awsLambda.deleteFunction(request);
        System.out.println("The " + functionName + " function was deleted");

    } catch (LambdaException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
}
```

- Per i dettagli sull'API, [DeleteFunction](#) consulta AWS SDK for Java 2.x API Reference.

## Invoke

Il seguente esempio di codice mostra come utilizzare `Invoke`.

## SDK per Java 2.x

 Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import org.json.JSONObject;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.services.lambda.LambdaClient;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.lambda.model.InvokeRequest;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.services.lambda.model.InvokeResponse;
import software.amazon.awssdk.services.lambda.model.LambdaException;

public class LambdaInvoke {

    /*
     * Function names appear as
     * arn:aws:lambda:us-west-2:335556666777:function:HelloFunction
     * you can retrieve the value by looking at the function in the AWS Console
     *
     * Also, set up your development environment, including your credentials.
     *
     * For information, see this documentation topic:
     *
     * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.
     * html
     */

    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <functionName>\s

            Where:
                functionName - The name of the Lambda function\s

            """;
```

```
    if (args.length != 1) {
        System.out.println(usage);
        System.exit(1);
    }

    String functionName = args[0];
    Region region = Region.US_WEST_2;
    LambdaClient awsLambda = LambdaClient.builder()
        .region(region)
        .build();

    invokeFunction(awsLambda, functionName);
    awsLambda.close();
}

public static void invokeFunction(LambdaClient awsLambda, String functionName) {

    InvokeResponse res = null;
    try {
        // Need a SdkBytes instance for the payload.
        JSONObject jsonObj = new JSONObject();
        jsonObj.put("inputValue", "2000");
        String json = jsonObj.toString();
        SdkBytes payload = SdkBytes.fromUtf8String(json);

        // Setup an InvokeRequest.
        InvokeRequest request = InvokeRequest.builder()
            .functionName(functionName)
            .payload(payload)
            .build();

        res = awsLambda.invoke(request);
        String value = res.payload().asUtf8String();
        System.out.println(value);

    } catch (LambdaException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
}
```

- Per informazioni dettagliate sulle API, consulta [Invoke](#) nella Documentazione di riferimento delle API AWS SDK for Java 2.x .

## Scenari

### Nozioni di base sulle funzioni

L'esempio di codice seguente mostra come:

- Crea un ruolo IAM e una funzione Lambda, quindi carica il codice del gestore.
- Richiamare la funzione con un singolo parametro e ottenere i risultati.
- Aggiorna il codice della funzione e configuralo con una variabile di ambiente.
- Richiamare la funzione con nuovi parametri e ottenere i risultati. Visualizza il log di esecuzione restituito.
- Elenca le funzioni dell'account, quindi elimina le risorse.

Per ulteriori informazioni sull'utilizzo di Lambda, consulta [Creare una funzione Lambda con la console](#).

### SDK per Java 2.x

#### Note

C'è dell'altro GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
/*
 * Lambda function names appear as:
 *
 * arn:aws:lambda:us-west-2:335556666777:function:HelloFunction
 *
 * To find this value, look at the function in the AWS Management Console.
 *
 * Before running this Java code example, set up your development environment,
 * including your credentials.
 *
 * For more information, see this documentation topic:
 *
```

```

* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*
* This example performs the following tasks:
*
* 1. Creates an AWS Lambda function.
* 2. Gets a specific AWS Lambda function.
* 3. Lists all Lambda functions.
* 4. Invokes a Lambda function.
* 5. Updates the Lambda function code and invokes it again.
* 6. Updates a Lambda function's configuration value.
* 7. Deletes a Lambda function.
*/

```

```

public class LambdaScenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    public static void main(String[] args) throws InterruptedException {
        final String usage = ""

            Usage:
                <functionName> <filePath> <role> <handler> <bucketName> <key>\s

            Where:
                functionName - The name of the Lambda function.\s
                filePath - The path to the .zip or .jar where the code is
located.\s
                role - The AWS Identity and Access Management (IAM) service role
that has Lambda permissions.\s
                handler - The fully qualified method name (for example,
example.Handler::handleRequest).\s
                bucketName - The Amazon Simple Storage Service (Amazon S3)
bucket name that contains the .zip or .jar used to update the Lambda function's
code.\s
                key - The Amazon S3 key name that represents the .zip or .jar
(for example, LambdaHello-1.0-SNAPSHOT.jar).
                """;

        if (args.length != 6) {
            System.out.println(usage);
            System.exit(1);
        }

        String functionName = args[0];
        String filePath = args[1];

```

```
String role = args[2];
String handler = args[3];
String bucketName = args[4];
String key = args[5];

Region region = Region.US_WEST_2;
LambdaClient awsLambda = LambdaClient.builder()
    .region(region)
    .build();

System.out.println(DASHES);
System.out.println("Welcome to the AWS Lambda example scenario.");
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("1. Create an AWS Lambda function.");
String funArn = createLambdaFunction(awsLambda, functionName, filePath,
role, handler);
System.out.println("The AWS Lambda ARN is " + funArn);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("2. Get the " + functionName + " AWS Lambda function.");
getFunction(awsLambda, functionName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("3. List all AWS Lambda functions.");
listFunctions(awsLambda);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("4. Invoke the Lambda function.");
System.out.println("*** Sleep for 1 min to get Lambda function ready.");
Thread.sleep(60000);
invokeFunction(awsLambda, functionName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("5. Update the Lambda function code and invoke it
again.");
updateFunctionCode(awsLambda, functionName, bucketName, key);
System.out.println("*** Sleep for 1 min to get Lambda function ready.");
Thread.sleep(60000);
```

```
        invokeFunction(awsLambda, functionName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6. Update a Lambda function's configuration value.");
        updateFunctionConfiguration(awsLambda, functionName, handler);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Delete the AWS Lambda function.");
        LambdaScenario.deleteLambdaFunction(awsLambda, functionName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("The AWS Lambda scenario completed successfully");
        System.out.println(DASHES);
        awsLambda.close();
    }

    public static String createLambdaFunction(LambdaClient awsLambda,
        String functionName,
        String filePath,
        String role,
        String handler) {

        try {
            LambdaWaiter waiter = awsLambda.waiter();
            InputStream is = new FileInputStream(filePath);
            SdkBytes fileToUpload = SdkBytes.fromInputStream(is);

            FunctionCode code = FunctionCode.builder()
                .zipFile(fileToUpload)
                .build();

            CreateFunctionRequest functionRequest = CreateFunctionRequest.builder()
                .functionName(functionName)
                .description("Created by the Lambda Java API")
                .code(code)
                .handler(handler)
                .runtime(Runtime.JAVA8)
                .role(role)
                .build();

            // Create a Lambda function using a waiter
```



```
        CreateFunctionResponse functionResponse =
awsLambda.createFunction(functionRequest);
        GetFunctionRequest getFunctionRequest = GetFunctionRequest.builder()
            .functionName(functionName)
            .build();
        WaiterResponse<GetFunctionResponse> waiterResponse =
waiter.waitUntilFunctionExists(getFunctionRequest);
        waiterResponse.matched().response().ifPresent(System.out::println);
        return functionResponse.functionArn();

    } catch (LambdaException | FileNotFoundException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return "";
}

public static void getFunction(LambdaClient awsLambda, String functionName) {
    try {
        GetFunctionRequest functionRequest = GetFunctionRequest.builder()
            .functionName(functionName)
            .build();

        GetFunctionResponse response = awsLambda.getFunction(functionRequest);
        System.out.println("The runtime of this Lambda function is " +
response.configuration().runtime());

    } catch (LambdaException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void listFunctions(LambdaClient awsLambda) {
    try {
        ListFunctionsResponse functionResult = awsLambda.listFunctions();
        List<FunctionConfiguration> list = functionResult.functions();
        for (FunctionConfiguration config : list) {
            System.out.println("The function name is " + config.functionName());
        }

    } catch (LambdaException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

```
    }  
  }  
  
  public static void invokeFunction(LambdaClient awsLambda, String functionName) {  
  
    InvokeResponse res;  
    try {  
      // Need a SdkBytes instance for the payload.  
      JSONObject jsonObj = new JSONObject();  
      jsonObj.put("inputValue", "2000");  
      String json = jsonObj.toString();  
      SdkBytes payload = SdkBytes.fromUtf8String(json);  
  
      InvokeRequest request = InvokeRequest.builder()  
        .functionName(functionName)  
        .payload(payload)  
        .build();  
  
      res = awsLambda.invoke(request);  
      String value = res.payload().asUtf8String();  
      System.out.println(value);  
  
    } catch (LambdaException e) {  
      System.err.println(e.getMessage());  
      System.exit(1);  
    }  
  }  
  
  public static void updateFunctionCode(LambdaClient awsLambda, String  
functionName, String bucketName, String key) {  
    try {  
      LambdaWaiter waiter = awsLambda.waiter();  
      UpdateFunctionCodeRequest functionCodeRequest =  
UpdateFunctionCodeRequest.builder()  
        .functionName(functionName)  
        .publish(true)  
        .s3Bucket(bucketName)  
        .s3Key(key)  
        .build();  
  
      UpdateFunctionCodeResponse response =  
awsLambda.updateFunctionCode(functionCodeRequest);  
      GetFunctionConfigurationRequest getFunctionConfigRequest =  
GetFunctionConfigurationRequest.builder()
```

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

        WaiterResponse<GetFunctionConfigurationResponse> waiterResponse = waiter
            .waitUntilFunctionUpdated(getFunctionConfigRequest);
        waiterResponse.matched().response().ifPresent(System.out::println);
        System.out.println("The last modified value is " +
response.lastModified());

    } catch (LambdaException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void updateFunctionConfiguration(LambdaClient awsLambda, String
functionName, String handler) {
    try {
        UpdateFunctionConfigurationRequest configurationRequest =
UpdateFunctionConfigurationRequest.builder()
            .functionName(functionName)
            .handler(handler)
            .runtime(Runtime.JAVA11)
            .build();

        awsLambda.updateFunctionConfiguration(configurationRequest);

    } catch (LambdaException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void deleteLambdaFunction(LambdaClient awsLambda, String
functionName) {
    try {
        DeleteFunctionRequest request = DeleteFunctionRequest.builder()
            .functionName(functionName)
            .build();

        awsLambda.deleteFunction(request);
        System.out.println("The " + functionName + " function was deleted");

    } catch (LambdaException e) {
```

```
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
}
```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella Documentazione di riferimento delle API AWS SDK for Java 2.x .
  - [CreateFunction](#)
  - [DeleteFunction](#)
  - [GetFunction](#)
  - [Invoke](#)
  - [ListFunctions](#)
  - [UpdateFunctionCode](#)
  - [UpdateFunctionConfiguration](#)

## Esempi serverless

### Richiamare una funzione Lambda da un trigger Kinesis

Il seguente esempio di codice mostra come implementare una funzione Lambda che riceve un evento attivato dalla ricezione di record da un flusso Kinesis. La funzione recupera il payload Kinesis, lo decodifica da Base64 e registra il contenuto del record.

### SDK per Java 2.x

#### Note

C'è altro su [GitHub](#) Trova l'esempio completo e scopri come eseguire la configurazione e l'esecuzione nel repository di [Esempi serverless](#).

### Utilizzo di un evento Kinesis con Lambda tramite Java.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
```

```
package example;

import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.LambdaLogger;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.KinesisEvent;

public class Handler implements RequestHandler<KinesisEvent, Void> {
    @Override
    public Void handleRequest(final KinesisEvent event, final Context context) {
        LambdaLogger logger = context.getLogger();
        if (event.getRecords().isEmpty()) {
            logger.log("Empty Kinesis Event received");
            return null;
        }
        for (KinesisEvent.KinesisEventRecord record : event.getRecords()) {
            try {
                logger.log("Processed Event with EventId: "+record.getEventID());
                String data = new String(record.getKinesis().getData().array());
                logger.log("Data:"+ data);
                // TODO: Do interesting work based on the new data
            }
            catch (Exception ex) {
                logger.log("An error occurred:"+ex.getMessage());
                throw ex;
            }
        }
        logger.log("Successfully processed:"+event.getRecords().size()+" records");
        return null;
    }
}
```

## Richiamo di una funzione Lambda da un trigger Amazon S3

Il seguente esempio di codice mostra come implementare una funzione Lambda che riceve un evento attivato dal caricamento di un oggetto in un bucket S3. La funzione recupera il nome del bucket S3 e la chiave dell'oggetto dal parametro evento e chiama l'API Amazon S3 per recuperare e registrare il tipo di contenuto dell'oggetto.

## SDK per Java 2.x

 Note

C'è di più su. GitHub Trova l'esempio completo e scopri come eseguire la configurazione e l'esecuzione nel repository di [Esempi serverless](#).

## Utilizzo di un evento S3 con Lambda tramite Java.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package example;

import software.amazon.awssdk.services.s3.model.HeadObjectRequest;
import software.amazon.awssdk.services.s3.model.HeadObjectResponse;
import software.amazon.awssdk.services.s3.S3Client;

import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.S3Event;
import
    com.amazonaws.services.lambda.runtime.events.models.s3.S3EventNotification.S3EventNotificat

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

public class Handler implements RequestHandler<S3Event, String> {
    private static final Logger logger = LoggerFactory.getLogger(Handler.class);
    @Override
    public String handleRequest(S3Event s3event, Context context) {
        try {
            S3EventNotificationRecord record = s3event.getRecords().get(0);
            String srcBucket = record.getS3().getBucket().getName();
            String srcKey = record.getS3().getObject().getUrlDecodedKey();

            S3Client s3Client = S3Client.builder().build();
            HeadObjectResponse headObject = getHeadObject(s3Client, srcBucket,
srcKey);

            logger.info("Successfully retrieved " + srcBucket + "/" + srcKey + " of
type " + headObject.contentType());
```

```
        return "Ok";
    } catch (Exception e) {
        throw new RuntimeException(e);
    }
}

private HeadObjectResponse getHeadObject(S3Client s3Client, String bucket,
String key) {
    HeadObjectRequest headObjectRequest = HeadObjectRequest.builder()
        .bucket(bucket)
        .key(key)
        .build();
    return s3Client.headObject(headObjectRequest);
}
}
```

## Richiamo di una funzione Lambda da un trigger Amazon SNS

Il seguente esempio di codice mostra come implementare una funzione Lambda che riceve un evento attivato dalla ricezione di messaggi da un argomento SNS. La funzione recupera i messaggi dal parametro dell'evento e registra il contenuto di ogni messaggio.

### SDK per Java 2.x

#### Note

C'è altro su [GitHub](#) Trova l'esempio completo e scopri come eseguire la configurazione e l'esecuzione nel repository di [Esempi serverless](#).

## Consumo di un evento SNS con Lambda utilizzando Java.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package example;

import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.LambdaLogger;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.SNSEvent;
import com.amazonaws.services.lambda.runtime.events.SNSEvent.SNSRecord;
```

```
import java.util.Iterator;
import java.util.List;

public class SNSEventHandler implements RequestHandler<SNSEvent, Boolean> {
    LambdaLogger logger;

    @Override
    public Boolean handleRequest(SNSEvent event, Context context) {
        logger = context.getLogger();
        List<SNSRecord> records = event.getRecords();
        if (!records.isEmpty()) {
            Iterator<SNSRecord> recordsIter = records.iterator();
            while (recordsIter.hasNext()) {
                processRecord(recordsIter.next());
            }
        }
        return Boolean.TRUE;
    }

    public void processRecord(SNSRecord record) {
        try {
            String message = record.getSNS().getMessage();
            logger.log("message: " + message);
        } catch (Exception e) {
            throw new RuntimeException(e);
        }
    }
}
```

## Richiamo di una funzione Lambda da un trigger Amazon SQS

Il seguente esempio di codice mostra come implementare una funzione Lambda che riceve un evento attivato dalla ricezione di messaggi da una coda SQS. La funzione recupera i messaggi dal parametro dell'evento e registra il contenuto di ogni messaggio.



## SDK per Java 2.x

 Note

C'è altro su GitHub Trova l'esempio completo e scopri come eseguire la configurazione e l'esecuzione nel repository di [Esempi serverless](#).

## Utilizzo di un evento SQS con Lambda tramite Java.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.SQSEvent;
import com.amazonaws.services.lambda.runtime.events.SQSEvent.SQSMessage;

public class Function implements RequestHandler<SQSEvent, Void> {
    @Override
    public Void handleRequest(SQSEvent sqsEvent, Context context) {
        for (SQSMessage msg : sqsEvent.getRecords()) {
            processMessage(msg, context);
        }
        context.getLogger().log("done");
        return null;
    }

    private void processMessage(SQSMessage msg, Context context) {
        try {
            context.getLogger().log("Processed message " + msg.getBody());

            // TODO: Do interesting work based on the new message

        } catch (Exception e) {
            context.getLogger().log("An error occurred");
            throw e;
        }
    }
}
```

## Segnalazione di errori di elementi batch per funzioni Lambda con un trigger Kinesis

Il seguente esempio di codice mostra come implementare una risposta batch parziale per le funzioni Lambda che ricevono eventi da un flusso Kinesis. La funzione riporta gli errori degli elementi batch nella risposta, segnalando a Lambda di riprovare tali messaggi in un secondo momento.

SDK per Java 2.x

### Note

C'è di più su [GitHub](#) Trova l'esempio completo e scopri come eseguire la configurazione e l'esecuzione nel repository di [Esempi serverless](#).

Segnalazione di errori di elementi batch di Kinesis con Lambda tramite Java.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.KinesisEvent;
import com.amazonaws.services.lambda.runtime.events.StreamsEventResponse;

import java.io.Serializable;
import java.util.ArrayList;
import java.util.List;

public class ProcessKinesisRecords implements RequestHandler<KinesisEvent,
StreamsEventResponse> {

    @Override
    public StreamsEventResponse handleRequest(KinesisEvent input, Context context) {

        List<StreamsEventResponse.BatchItemFailure> batchItemFailures = new
ArrayList<>();
        String curRecordSequenceNumber = "";

        for (KinesisEvent.KinesisEventRecord kinesisEventRecord :
input.getRecords()) {
            try {
                //Process your record
                KinesisEvent.Record kinesisRecord = kinesisEventRecord.getKinesis();
```

```

        curRecordSequenceNumber = kinesisRecord.getSequenceNumber();

    } catch (Exception e) {
        /* Since we are working with streams, we can return the failed item
immediately.
        Lambda will immediately begin to retry processing from this
failed item onwards. */
        batchItemFailures.add(new
StreamsEventResponse.BatchItemFailure(curRecordSequenceNumber));
        return new StreamsEventResponse(batchItemFailures);
    }
}

return new StreamsEventResponse(batchItemFailures);
}
}

```

Segnalazione degli errori degli elementi batch per le funzioni Lambda con un trigger DynamoDB

Il seguente esempio di codice mostra come implementare una risposta batch parziale per le funzioni Lambda che ricevono eventi da un flusso DynamoDB. La funzione riporta gli errori degli elementi batch nella risposta, segnalando a Lambda di riprovare tali messaggi in un secondo momento.

SDK per Java 2.x

#### Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri come eseguire la configurazione e l'esecuzione nel repository di [Esempi serverless](#).

Segnalazione degli errori degli elementi batch di DynamoDB con Lambda utilizzando Java.

```

// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.DynamodbEvent;
import com.amazonaws.services.lambda.runtime.events.StreamsEventResponse;
import com.amazonaws.services.lambda.runtime.events.models.dynamodb.StreamRecord;

```

```
import java.io.Serializable;
import java.util.ArrayList;
import java.util.List;

public class ProcessDynamodbRecords implements RequestHandler<DynamodbEvent,
    Serializable> {

    @Override
    public StreamsEventResponse handleRequest(DynamodbEvent input, Context context)
    {

        List<StreamsEventResponse.BatchItemFailure> batchItemFailures = new
        ArrayList<>();
        String curRecordSequenceNumber = "";

        for (DynamodbEvent.DynamodbStreamRecord dynamodbStreamRecord :
        input.getRecords()) {
            try {
                //Process your record
                StreamRecord dynamodbRecord = dynamodbStreamRecord.getDynamodb();
                curRecordSequenceNumber = dynamodbRecord.getSequenceNumber();

            } catch (Exception e) {
                /* Since we are working with streams, we can return the failed item
                immediately.
                Lambda will immediately begin to retry processing from this
                failed item onwards. */
                batchItemFailures.add(new
                StreamsEventResponse.BatchItemFailure(curRecordSequenceNumber));
                return new StreamsEventResponse(batchItemFailures);
            }
        }

        return new StreamsEventResponse();
    }
}
```

## Segnalazione di errori di elementi batch per funzioni Lambda con un trigger Amazon SQS

Il seguente esempio di codice mostra come implementare una risposta batch parziale per le funzioni Lambda che ricevono eventi da una coda SQS. La funzione riporta gli errori degli elementi batch nella risposta, segnalando a Lambda di riprovare tali messaggi in un secondo momento.

SDK per Java 2.x

### Note

C'è di più su [GitHub](#) Trova l'esempio completo e scopri come eseguire la configurazione e l'esecuzione nel repository di [Esempi serverless](#).

Segnalazione di errori di elementi batch di SQS con Lambda tramite Java.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.SQSEvent;
import com.amazonaws.services.lambda.runtime.events.SQSBatchResponse;

import java.util.ArrayList;
import java.util.List;

public class ProcessSQSMessageBatch implements RequestHandler<SQSEvent,
SQSBatchResponse> {
    @Override
    public SQSBatchResponse handleRequest(SQSEvent sqsEvent, Context context) {

        List<SQSBatchResponse.BatchItemFailure> batchItemFailures = new
ArrayList<SQSBatchResponse.BatchItemFailure>();
        String messageId = "";
        for (SQSEvent.SQSMessage message : sqsEvent.getRecords()) {
            try {
                //process your message
                messageId = message.getMessageId();
            } catch (Exception e) {
                //Add failed message identifier to the batchItemFailures list
                batchItemFailures.add(new
SQSBatchResponse.BatchItemFailure(messageId));
            }
        }
    }
}
```

```
        }  
    }  
    return new SQSBatchResponse(batchItemFailures);  
}  
}
```

## MediaConvert esempi che utilizzano SDK for Java 2.x

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando AWS SDK for Java 2.x with MediaConvert.

Le operazioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le operazioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati e negli esempi tra servizi.

Scenari: esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio.

Ogni esempio include un collegamento a GitHub, dove è possibile trovare istruzioni su come configurare ed eseguire il codice nel contesto.

### Argomenti

- [Azioni](#)

### Azioni

#### **CreateJob**

Il seguente esempio di codice mostra come utilizzare `CreateJob`.

SDK per Java 2.x

#### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
package com.example.mediaconvert;
```

```
import java.net.URI;
import java.util.HashMap;
import java.util.Map;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.mediaconvert.MediaConvertClient;
import software.amazon.awssdk.services.mediaconvert.model.DescribeEndpointsResponse;
import software.amazon.awssdk.services.mediaconvert.model.DescribeEndpointsRequest;
import software.amazon.awssdk.services.mediaconvert.model.Output;
import software.amazon.awssdk.services.mediaconvert.model.MediaConvertException;
import software.amazon.awssdk.services.mediaconvert.model.OutputGroup;
import software.amazon.awssdk.services.mediaconvert.model.OutputGroupSettings;
import software.amazon.awssdk.services.mediaconvert.model.HlsGroupSettings;
import software.amazon.awssdk.services.mediaconvert.model.OutputGroupType;
import software.amazon.awssdk.services.mediaconvert.model.HlsDirectoryStructure;
import software.amazon.awssdk.services.mediaconvert.model.HlsManifestDurationFormat;
import software.amazon.awssdk.services.mediaconvert.model.HlsStreamInfResolution;
import software.amazon.awssdk.services.mediaconvert.model.HlsClientCache;
import software.amazon.awssdk.services.mediaconvert.model.HlsCaptionLanguageSetting;
import software.amazon.awssdk.services.mediaconvert.model.HlsManifestCompression;
import software.amazon.awssdk.services.mediaconvert.model.HlsCodecSpecification;
import software.amazon.awssdk.services.mediaconvert.model.HlsOutputSelection;
import software.amazon.awssdk.services.mediaconvert.model.HlsProgramDateTime;
import software.amazon.awssdk.services.mediaconvert.model.HlsTimedMetadataId3Frame;
import software.amazon.awssdk.services.mediaconvert.model.HlsSegmentControl;
import software.amazon.awssdk.services.mediaconvert.model.FileGroupSettings;
import software.amazon.awssdk.services.mediaconvert.model.ContainerSettings;
import software.amazon.awssdk.services.mediaconvert.model.VideoDescription;
import software.amazon.awssdk.services.mediaconvert.model.ContainerType;
import software.amazon.awssdk.services.mediaconvert.model.ScalingBehavior;
import software.amazon.awssdk.services.mediaconvert.model.VideoTimecodeInsertion;
import software.amazon.awssdk.services.mediaconvert.model.ColorMetadata;
import software.amazon.awssdk.services.mediaconvert.model.RespondToAfd;
import software.amazon.awssdk.services.mediaconvert.model.AfdSignaling;
import software.amazon.awssdk.services.mediaconvert.model.DropFrameTimecode;
import software.amazon.awssdk.services.mediaconvert.model.VideoCodecSettings;
import software.amazon.awssdk.services.mediaconvert.model.H264Settings;
import software.amazon.awssdk.services.mediaconvert.model.VideoCodec;
import software.amazon.awssdk.services.mediaconvert.model.CreateJobRequest;
import software.amazon.awssdk.services.mediaconvert.model.H264RateControlMode;
import software.amazon.awssdk.services.mediaconvert.model.H264QualityTuningLevel;
import software.amazon.awssdk.services.mediaconvert.model.H264SceneChangeDetect;
import
software.amazon.awssdk.services.mediaconvert.model.AacAudioDescriptionBroadcasterMix;
```

```
import software.amazon.awssdk.services.mediaconvert.model.H264ParControl;
import software.amazon.awssdk.services.mediaconvert.model.AacRawFormat;
import software.amazon.awssdk.services.mediaconvert.model.H264QvbrSettings;
import
    software.amazon.awssdk.services.mediaconvert.model.H264FramerateConversionAlgorithm;
import software.amazon.awssdk.services.mediaconvert.model.H264CodecLevel;
import software.amazon.awssdk.services.mediaconvert.model.H264FramerateControl;
import software.amazon.awssdk.services.mediaconvert.model.AacCodingMode;
import software.amazon.awssdk.services.mediaconvert.model.H264Telecine;
import
    software.amazon.awssdk.services.mediaconvert.model.H264FlickerAdaptiveQuantization;
import software.amazon.awssdk.services.mediaconvert.model.H264GopSizeUnits;
import software.amazon.awssdk.services.mediaconvert.model.H264CodecProfile;
import software.amazon.awssdk.services.mediaconvert.model.H264GopBReference;
import software.amazon.awssdk.services.mediaconvert.model.AudioTypeControl;
import software.amazon.awssdk.services.mediaconvert.model.AntiAlias;
import software.amazon.awssdk.services.mediaconvert.model.H264SlowPal;
import
    software.amazon.awssdk.services.mediaconvert.model.H264SpatialAdaptiveQuantization;
import software.amazon.awssdk.services.mediaconvert.model.H264Syntax;
import software.amazon.awssdk.services.mediaconvert.model.M3u8Settings;
import software.amazon.awssdk.services.mediaconvert.model.InputDenoiseFilter;
import
    software.amazon.awssdk.services.mediaconvert.model.H264TemporalAdaptiveQuantization;
import software.amazon.awssdk.services.mediaconvert.model.CreateJobResponse;
import
    software.amazon.awssdk.services.mediaconvert.model.H264UnregisteredSeiTimecode;
import software.amazon.awssdk.services.mediaconvert.model.H264EntropyEncoding;
import software.amazon.awssdk.services.mediaconvert.model.InputPsiControl;
import software.amazon.awssdk.services.mediaconvert.model.ColorSpace;
import software.amazon.awssdk.services.mediaconvert.model.H264RepeatPps;
import software.amazon.awssdk.services.mediaconvert.model.H264FieldEncoding;
import software.amazon.awssdk.services.mediaconvert.model.M3u8NielsenId3;
import software.amazon.awssdk.services.mediaconvert.model.InputDeblockFilter;
import software.amazon.awssdk.services.mediaconvert.model.InputRotate;
import software.amazon.awssdk.services.mediaconvert.model.H264DynamicSubGop;
import software.amazon.awssdk.services.mediaconvert.model.TimedMetadata;
import software.amazon.awssdk.services.mediaconvert.model.JobSettings;
import software.amazon.awssdk.services.mediaconvert.model.AudioDefaultSelection;
import software.amazon.awssdk.services.mediaconvert.model.VideoSelector;
import software.amazon.awssdk.services.mediaconvert.model.AacSpecification;
import software.amazon.awssdk.services.mediaconvert.model.Input;
import software.amazon.awssdk.services.mediaconvert.model.OutputSettings;
import software.amazon.awssdk.services.mediaconvert.model.H264AdaptiveQuantization;
```



```

import software.amazon.awssdk.services.mediaconvert.model.AudioLanguageCodeControl;
import software.amazon.awssdk.services.mediaconvert.model.InputFilterEnable;
import software.amazon.awssdk.services.mediaconvert.model.AudioDescription;
import software.amazon.awssdk.services.mediaconvert.model.H264InterlaceMode;
import software.amazon.awssdk.services.mediaconvert.model.AudioCodecSettings;
import software.amazon.awssdk.services.mediaconvert.model.AacSettings;
import software.amazon.awssdk.services.mediaconvert.model.AudioCodec;
import software.amazon.awssdk.services.mediaconvert.model.AacRateControlMode;
import software.amazon.awssdk.services.mediaconvert.model.AacCodecProfile;
import software.amazon.awssdk.services.mediaconvert.model.HlsIFrameOnlyManifest;
import software.amazon.awssdk.services.mediaconvert.model.FrameCaptureSettings;
import software.amazon.awssdk.services.mediaconvert.model.AudioSelector;
import software.amazon.awssdk.services.mediaconvert.model.M3u8PcrControl;
import software.amazon.awssdk.services.mediaconvert.model.InputTimecodeSource;
import software.amazon.awssdk.services.mediaconvert.model.HlsSettings;
import software.amazon.awssdk.services.mediaconvert.model.M3u8Scte35Source;

/**
 * Create a MediaConvert job. Must supply MediaConvert access role Amazon
 * Resource Name (ARN), and a
 * valid video input file via Amazon S3 URL.
 *
 * Also, set up your development environment, including your credentials.
 *
 * For information, see this documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 */
public class CreateJob {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <mcRoleARN> <fileInput>\s

                Where:
                mcRoleARN - The MediaConvert Role ARN.\s
                fileInput - The URL of an Amazon S3 bucket
                where the input file is located.\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
        }
    }
}

```

```
        System.exit(1);
    }

    String mcRoleARN = args[0];
    String fileInput = args[1];
    Region region = Region.US_WEST_2;
    MediaConvertClient mc = MediaConvertClient.builder()
        .region(region)
        .build();

    String id = createMediaJob(mc, mcRoleARN, fileInput);
    System.out.println("MediaConvert job created. Job Id = " + id);
    mc.close();
}

public static String createMediaJob(MediaConvertClient mc, String mcRoleARN,
String fileInput) {

    String s3path = fileInput.substring(0, fileInput.lastIndexOf('/') +
1) + "javasdk/out/";
    String fileOutput = s3path + "index";
    String thumbsOutput = s3path + "thumbs/";
    String mp4Output = s3path + "mp4/";

    try {
        DescribeEndpointsResponse res = mc

.describeEndpoints(DescribeEndpointsRequest.builder().maxResults(20).build());

        if (res.endpoints().size() <= 0) {
            System.out.println("Cannot find MediaConvert service
endpoint URL!");

            System.exit(1);
        }
        String endpointURL = res.endpoints().get(0).url();
        System.out.println("MediaConvert service URL: " +
endpointURL);

        System.out.println("MediaConvert role arn: " + mcRoleARN);
        System.out.println("MediaConvert input file: " + fileInput);
        System.out.println("MediaConvert output path: " + s3path);

        MediaConvertClient emc = MediaConvertClient.builder()
            .region(Region.US_WEST_2)
            .endpointOverride(URI.create(endpointURL))
```

```
        .build();

        // output group Preset HLS low profile
        Output hlsLow = createOutput("hls_low", "_low", "_$dt$",
750000, 7, 1920, 1080, 640);
        // output group Preset HLS media profile
        Output hlsMedium = createOutput("hls_medium", "_medium", "_
$dt$", 1200000, 7, 1920, 1080, 1280);
        // output group Preset HLS high profole
        Output hlsHigh = createOutput("hls_high", "_high", "_$dt$",
3500000, 8, 1920, 1080, 1920);

        OutputGroup appleHLS = OutputGroup.builder().name("Apple
HLS").customName("Example")

        .outputGroupSettings(OutputGroupSettings.builder()

        .type(OutputGroupType.HLS_GROUP_SETTINGS)

        .hlsGroupSettings(HlsGroupSettings.builder()

        .directoryStructure(

            HlsDirectoryStructure.SINGLE_DIRECTORY)

        .manifestDurationFormat(

            HlsManifestDurationFormat.INTEGER)

        .streamInfResolution(

            HlsStreamInfResolution.INCLUDE)

        .clientCache(HlsClientCache.ENABLED)

        .captionLanguageSetting(

            HlsCaptionLanguageSetting.OMIT)

        .manifestCompression(

            HlsManifestCompression.NONE)

        .codecSpecification(
```

```

        HlsCodecSpecification.RFC_4281)

    .outputSelection(

        HlsOutputSelection.MANIFESTS_AND_SEGMENTS)

    .programDateTime(HlsProgramDateTime.EXCLUDE)

    .programDateTimePeriod(600)

    .timedMetadataId3Frame(

        HlsTimedMetadataId3Frame.PRIV)

    .timedMetadataId3Period(10)

    .destination(fileOutput)

    .segmentControl(HlsSegmentControl.SEGMENTED_FILES)

    .minFinalSegmentLength((double) 0)

    .segmentLength(4).minSegmentLength(0).build()
                                                                    .build()
                                                                    .outputs(hlsLow, hlsMedium,
hlsHigh).build();

        OutputGroup fileMp4 = OutputGroup.builder().name("File
Group").customName("mp4")

    .outputGroupSettings(OutputGroupSettings.builder()

    .type(OutputGroupType.FILE_GROUP_SETTINGS)

    .fileGroupSettings(FileGroupSettings.builder()

    .destination(mp4Output).build()
                                                                    .build()
                                                                    .outputs(Output.builder().extension("mp4")

    .containerSettings(ContainerSettings.builder()

    .container(ContainerType.MP4).build())

```

```
.videoDescription(VideoDescription.builder().width(1280)
                                                    .height(720)
                                                    .scalingBehavior(ScalingBehavior.DEFAULT)
                                                    .sharpness(50).antiAlias(AntiAlias.ENABLED)
                                                    .timecodeInsertion(
                VideoTimecodeInsertion.DISABLED)
                                                    .colorMetadata(ColorMetadata.INSERT)
                                                    .respondToAfd(RespondToAfd.NONE)
                                                    .afdSignaling(AfdSignaling.NONE)
                                                    .dropFrameTimecode(DropFrameTimecode.ENABLED)
                                                    .codecSettings(VideoCodecSettings.builder()
                .codec(VideoCodec.H_264)
                .h264Settings(H264Settings
                    .builder()
                    .rateControlMode(
                        H264RateControlMode.QVBR)
                    .parControl(H264ParControl.INITIALIZE_FROM_SOURCE)
                    .qualityTuningLevel(
                        H264QualityTuningLevel.SINGLE_PASS)
                    .qvbrSettings(
                        H264QvbrSettings.builder()
                            .qvbrQualityLevel(
```

```
8)

        .build()

        .codecLevel(H264CodecLevel.AUTO)

        .codecProfile(H264CodecProfile.MAIN)

        .maxBitrate(2400000)

        .framerateControl(

            H264FramerateControl.INITIALIZE_FROM_SOURCE)

        .gopSize(2.0)

        .gopSizeUnits(H264GopSizeUnits.SECONDS)

        .numberBFramesBetweenReferenceFrames(

            2)

        .gopClosedCadence(

            1)

        .gopBReference(H264GopBReference.DISABLED)

        .slowPal(H264SlowPal.DISABLED)

        .syntax(H264Syntax.DEFAULT)

        .numberReferenceFrames(

            3)

        .dynamicSubGop(H264DynamicSubGop.STATIC)

        .fieldEncoding(H264FieldEncoding.PAFF)

        .sceneChangeDetect(

            H264SceneChangeDetect.ENABLED)
```

```
.minIInterval(0)

.telecine(H264Telecine.NONE)

.framerateConversionAlgorithm(
    H264FramerateConversionAlgorithm.DUPLICATE_DROP)

.entropyEncoding(
    H264EntropyEncoding.CABAC)

.slices(1)

.unregisteredSeiTimecode(
    H264UnregisteredSeiTimecode.DISABLED)

.repeatPps(H264RepeatPps.DISABLED)

.adaptiveQuantization(
    H264AdaptiveQuantization.HIGH)

.spatialAdaptiveQuantization(
    H264SpatialAdaptiveQuantization.ENABLED)

.temporalAdaptiveQuantization(
    H264TemporalAdaptiveQuantization.ENABLED)

.flickerAdaptiveQuantization(
    H264FlickerAdaptiveQuantization.DISABLED)

.softness(0)

.interlaceMode(H264InterlaceMode.PROGRESSIVE)

.build()

.build()
```

```

                                                                                          .build())

.audioDescriptions(AudioDescription.builder()

.audioTypeControl(AudioTypeControl.FOLLOW_INPUT)

.languageCodeControl(
    AudioLanguageCodeControl.FOLLOW_INPUT)

.codecSettings(AudioCodecSettings.builder()

    .codec(AudioCodec.AAC)

    .aacSettings(AacSettings
        .builder()
        .codecProfile(AacCodecProfile.LC)
        .rateControlMode(
            AacRateControlMode.CBR)
        .codingMode(AacCodingMode.CODING_MODE_2_0)
        .sampleRate(44100)
        .bitrate(160000)
        .rawFormat(AacRawFormat.NONE)
        .specification(AacSpecification.MPEG4)
        .audioDescriptionBroadcasterMix(
            AacAudioDescriptionBroadcasterMix.NORMAL)
        .build())

    .build())

                                                                                          .build())

                                                                                          .build();
```



```
OutputGroup thumbs = OutputGroup.builder().name("File
Group").customName("thumbs")

.outputGroupSettings(OutputGroupSettings.builder()

.type(OutputGroupType.FILE_GROUP_SETTINGS)

.fileGroupSettings(FileGroupSettings.builder()

.destination(thumbsOutput).build())

                                .build())
                                .outputs(Output.builder().extension("jpg")

.containerSettings(ContainerSettings.builder()

.container(ContainerType.RAW).build())

.videoDescription(VideoDescription.builder()

.scalingBehavior(ScalingBehavior.DEFAULT)

.sharpness(50).antiAlias(AntiAlias.ENABLED)

.timecodeInsertion(

    VideoTimecodeInsertion.DISABLED)

.colorMetadata(ColorMetadata.INSERT)

.dropFrameTimecode(DropFrameTimecode.ENABLED)

.codecSettings(VideoCodecSettings.builder()

    .codec(VideoCodec.FRAME_CAPTURE)

    .frameCaptureSettings(

        FrameCaptureSettings

            .builder()

                .framerateNumerator(

                    1)
```

```

        .framerateDenominator(
            1)
        .maxCaptures(10000000)
        .quality(80)
        .build()

    .build()

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

    Map<String, AudioSelector> audioSelectors = new HashMap<>();
    audioSelectors.put("Audio Selector 1",
        AudioSelector.builder().defaultSelection(AudioDefaultSelection.DEFAULT)
            .offset(0).build());

    JobSettings jobSettings =
    JobSettings.builder().inputs(Input.builder()
        .audioSelectors(audioSelectors)
        .videoSelector(
            VideoSelector.builder().colorSpace(ColorSpace.FOLLOW)
                .rotate(InputRotate.DEGREE_0).build())
        .filterEnable(InputFilterEnable.AUTO).filterStrength(0)
            .deblockFilter(InputDeblockFilter.DISABLED)
        .denoiseFilter(InputDenoiseFilter.DISABLED).psiControl(InputPsiControl.USE_PSI)
        .timecodeSource(InputTimecodeSource.EMBEDDED).fileInput(fileInput).build())
        .outputGroups(appleHLS, thumbs,
            fileMp4).build());

    CreateJobRequest createJobRequest =
    CreateJobRequest.builder().role(mcRoleARN)
        .settings(jobSettings)
        .build();

```

```

        CreateJobResponse createJobResponse =
emc.createJob(createJobRequest);
        return createJobResponse.job().id();

    } catch (MediaConvertException e) {
        System.out.println(e.toString());
        System.exit(0);
    }
    return "";
}

private final static Output createOutput(String customName,
        String nameModifier,
        String segmentModifier,
        int qvbrMaxBitrate,
        int qvbrQualityLevel,
        int originWidth,
        int originHeight,
        int targetWidth) {

    int targetHeight = Math.round(originHeight * targetWidth /
originWidth)
        - (Math.round(originHeight * targetWidth /
originWidth) % 4);
    Output output = null;
    try {
        output =
Output.builder().nameModifier(nameModifier).outputSettings(OutputSettings.builder()

.hlsSettings(HlsSettings.builder().segmentModifier(segmentModifier)

.audioGroupId("program_audio")

.iFrameOnlyManifest(HlsIFrameOnlyManifest.EXCLUDE).build())
        .build())

.containerSettings(ContainerSettings.builder().container(ContainerType.M3_U8)

.m3u8Settings(M3u8Settings.builder().audioFramesPerPes(4)

.pcrControl(M3u8PcrControl.PCR_EVERY_PES_PACKET)

.pmtPid(480).privateMetadataPid(503)

```

```

.programNumber(1).patInterval(0).pmtInterval(0)

.scte35Source(M3u8Scte35Source.NONE)

.scte35Pid(500).nielsenId3(M3u8NielsenId3.NONE)

.timedMetadata(TimedMetadata.NONE)

.timedMetadataPid(502).videoPid(481)

.audioPids(482, 483, 484, 485, 486, 487, 488,
           489, 490, 491, 492)

                                           .build()
                                           .build()
                                           .videoDescription(
VideoDescription.builder().width(targetWidth)

.height(targetHeight)

.scalingBehavior(ScalingBehavior.DEFAULT)

.sharpness(50).antiAlias(AntiAlias.ENABLED)

.timecodeInsertion(
    VideoTimecodeInsertion.DISABLED)

.colorMetadata(ColorMetadata.INSERT)

.respondToAfd(RespondToAfd.NONE)

.afdSignaling(AfdSignaling.NONE)

.dropFrameTimecode(DropFrameTimecode.ENABLED)

.codecSettings(VideoCodecSettings.builder()

    .codec(VideoCodec.H_264)

    .h264Settings(H264Settings

```

```
.builder()

.rateControlMode(
    H264RateControlMode.QVBR)

.parControl(H264ParControl.INITIALIZE_FROM_SOURCE)

.qualityTuningLevel(
    H264QualityTuningLevel.SINGLE_PASS)

.qvbrSettings(H264QvbrSettings
    .builder()
    .qvbrQualityLevel(
        qvbrQualityLevel)
    .build())

.codecLevel(H264CodecLevel.AUTO)

.codecProfile((targetHeight > 720
    && targetWidth > 1280)
    ? H264CodecProfile.HIGH
    : H264CodecProfile.MAIN)

.maxBitrate(qvbrMaxBitrate)

.framerateControl(
    H264FramerateControl.INITIALIZE_FROM_SOURCE)

.gopSize(2.0)

.gopSizeUnits(H264GopSizeUnits.SECONDS)

.numberBFramesBetweenReferenceFrames(
```

```
                2)
        .gopClosedCadence(
                1)
        .gopBReference(H264GopBReference.DISABLED)
        .slowPal(H264SlowPal.DISABLED)
        .syntax(H264Syntax.DEFAULT)
        .numberReferenceFrames(
                3)
        .dynamicSubGop(H264DynamicSubGop.STATIC)
        .fieldEncoding(H264FieldEncoding.PAFF)
        .sceneChangeDetect(
                H264SceneChangeDetect.ENABLED)
        .minIInterval(0)
        .telecine(H264Telecine.NONE)
        .framerateConversionAlgorithm(
                H264FramerateConversionAlgorithm.DUPLICATE_DROP)
        .entropyEncoding(
                H264EntropyEncoding.CABAC)
        .slices(1)
        .unregisteredSeiTimecode(
                H264UnregisteredSeiTimecode.DISABLED)
        .repeatPps(H264RepeatPps.DISABLED)
```

```
.adaptiveQuantization(  
    H264AdaptiveQuantization.HIGH)  
.spatialAdaptiveQuantization(  
    H264SpatialAdaptiveQuantization.ENABLED)  
.temporalAdaptiveQuantization(  
    H264TemporalAdaptiveQuantization.ENABLED)  
.flickerAdaptiveQuantization(  
    H264FlickerAdaptiveQuantization.DISABLED)  
.softness(0)  
.interlaceMode(H264InterlaceMode.PROGRESSIVE)  
.build()  
  
.build()                                     .build()  
  
.audioDescriptions(AudioDescription.builder())  
.audioTypeControl(AudioTypeControl.FOLLOW_INPUT)  
.languageCodeControl(AudioLanguageCodeControl.FOLLOW_INPUT)  
.codecSettings(AudioCodecSettings.builder())  
.codec(AudioCodec.AAC).aacSettings(AacSettings  
    .builder()  
    .codecProfile(AacCodecProfile.LC)  
    .rateControlMode(  
        AacRateControlMode.CBR)
```

```

        .codingMode(AacCodingMode.CODING_MODE_2_0)

        .sampleRate(44100)

        .bitrate(96000)

        .rawFormat(AacRawFormat.NONE)

        .specification(AacSpecification.MPEG4)

        .audioDescriptionBroadcasterMix(

            AacAudioDescriptionBroadcasterMix.NORMAL)

        .build()

        .build()

        .build()

        .build();
    } catch (MediaConvertException e) {
        e.printStackTrace();
        System.exit(0);
    }
    return output;
}
}

```

- Per i dettagli sull'API, [CreateJob](#) consulta AWS SDK for Java 2.x API Reference.

## GetJob

Il seguente esempio di codice mostra come utilizzare `GetJob`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).



```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.mediaconvert.model.DescribeEndpointsResponse;
import software.amazon.awssdk.services.mediaconvert.model.GetJobRequest;
import software.amazon.awssdk.services.mediaconvert.model.DescribeEndpointsRequest;
import software.amazon.awssdk.services.mediaconvert.model.GetJobResponse;
import software.amazon.awssdk.services.mediaconvert.model.MediaConvertException;
import software.amazon.awssdk.services.mediaconvert.MediaConvertClient;
import java.net.URI;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class GetJob {

    public static void main(String[] args) {

        final String usage = "\n" +
            " <jobId> \n\n" +
            "Where:\n" +
            " jobId - The job id value.\n\n";

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String jobId = args[0];
        Region region = Region.US_WEST_2;
        MediaConvertClient mc = MediaConvertClient.builder()
            .region(region)
            .build();

        getSpecificJob(mc, jobId);
        mc.close();
    }

    public static void getSpecificJob(MediaConvertClient mc, String jobId) {
        try {
```

```
DescribeEndpointsResponse res =
mc.describeEndpoints(DescribeEndpointsRequest.builder()
    .maxResults(20)
    .build());

if (res.endpoints().size() <= 0) {
    System.out.println("Cannot find MediaConvert service endpoint
URL!");
    System.exit(1);
}
String endpointURL = res.endpoints().get(0).url();
MediaConvertClient emc = MediaConvertClient.builder()
    .region(Region.US_WEST_2)
    .endpointOverride(URI.create(endpointURL))
    .build();

GetJobRequest jobRequest = GetJobRequest.builder()
    .id(jobId)
    .build();

GetJobResponse response = emc.getJob(jobRequest);
System.out.println("The ARN of the job is " + response.job().arn());

} catch (MediaConvertException e) {
    System.out.println(e.toString());
    System.exit(0);
}
}
```

- Per i dettagli sull'API, [GetJob](#) consulta AWS SDK for Java 2.x API Reference.

## ListJobs

Il seguente esempio di codice mostra come utilizzare `ListJobs`.

## SDK per Java 2.x

 Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.mediaconvert.MediaConvertClient;
import software.amazon.awssdk.services.mediaconvert.model.ListJobsRequest;
import software.amazon.awssdk.services.mediaconvert.model.DescribeEndpointsResponse;
import software.amazon.awssdk.services.mediaconvert.model.DescribeEndpointsRequest;
import software.amazon.awssdk.services.mediaconvert.model.ListJobsResponse;
import software.amazon.awssdk.services.mediaconvert.model.Job;
import software.amazon.awssdk.services.mediaconvert.model.MediaConvertException;
import java.net.URI;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListJobs {
    public static void main(String[] args) {
        Region region = Region.US_WEST_2;
        MediaConvertClient mc = MediaConvertClient.builder()
            .region(region)
            .build();

        listCompleteJobs(mc);
        mc.close();
    }

    public static void listCompleteJobs(MediaConvertClient mc) {
        try {
            DescribeEndpointsResponse res =
                mc.describeEndpoints(DescribeEndpointsRequest.builder()
```

```
        .maxResults(20)
        .build());

    if (res.endpoints().size() <= 0) {
        System.out.println("Cannot find MediaConvert service endpoint
URL!");
        System.exit(1);
    }

    String endpointURL = res.endpoints().get(0).url();
    MediaConvertClient emc = MediaConvertClient.builder()
        .region(Region.US_WEST_2)
        .endpointOverride(URI.create(endpointURL))
        .build();

    ListJobsRequest jobsRequest = ListJobsRequest.builder()
        .maxResults(10)
        .status("COMPLETE")
        .build();

    ListJobsResponse jobsResponse = emc.listJobs(jobsRequest);
    List<Job> jobs = jobsResponse.jobs();
    for (Job job : jobs) {
        System.out.println("The JOB ARN is : " + job.arn());
    }

} catch (MediaConvertException e) {
    System.out.println(e.toString());
    System.exit(0);
}
}
```

- Per i dettagli sull'API, [ListJobs](#) consulta AWS SDK for Java 2.x API Reference.

## Esempi di Migration Hub con SDK for Java 2.x

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando AWS SDK for Java 2.x with Migration Hub.

Le operazioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le operazioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati e negli esempi tra servizi.

Scenari: esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio.

Ogni esempio include un collegamento a GitHub, dove è possibile trovare istruzioni su come configurare ed eseguire il codice nel contesto.

## Argomenti

- [Azioni](#)

## Azioni

### DeleteProgressUpdateStream

Il seguente esempio di codice mostra come utilizzare DeleteProgressUpdateStream.

SDK per Java 2.x

#### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.migrationhub.MigrationHubClient;
import
    software.amazon.awssdk.services.migrationhub.model.DeleteProgressUpdateStreamRequest;
import software.amazon.awssdk.services.migrationhub.model.MigrationHubException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
```

```
public class DeleteProgressStream {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <progressStream>\s

            Where:
                progressStream - the name of a progress stream to delete.\s
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String progressStream = args[0];
        Region region = Region.US_WEST_2;
        MigrationHubClient migrationClient = MigrationHubClient.builder()
            .region(region)
            .build();

        deleteStream(migrationClient, progressStream);
        migrationClient.close();
    }

    public static void deleteStream(MigrationHubClient migrationClient, String
streamName) {
        try {
            DeleteProgressUpdateStreamRequest deleteProgressUpdateStreamRequest =
DeleteProgressUpdateStreamRequest
                .builder()
                .progressUpdateStreamName(streamName)
                .build();

            migrationClient.deleteProgressUpdateStream(deleteProgressUpdateStreamRequest);
            System.out.println(streamName + " is deleted");

        } catch (MigrationHubException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

```
}
```

- Per i dettagli sull'API, [DeleteProgressUpdateStream](#) consulta AWS SDK for Java 2.x API Reference.

## DescribeApplicationState

Il seguente esempio di codice mostra come utilizzare `DescribeApplicationState`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.migrationhub.MigrationHubClient;
import
    software.amazon.awssdk.services.migrationhub.model.DescribeApplicationStateRequest;
import
    software.amazon.awssdk.services.migrationhub.model.DescribeApplicationStateResponse;
import software.amazon.awssdk.services.migrationhub.model.MigrationHubException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DescribeAppState {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                DescribeAppState <appId>\s
```

```
        Where:
            appId - the application id value.\s
            """;

    if (args.length != 1) {
        System.out.println(usage);
        System.exit(1);
    }

    String appId = args[0];
    Region region = Region.US_WEST_2;
    MigrationHubClient migrationClient = MigrationHubClient.builder()
        .region(region)
        .build();

    describeApplicationState(migrationClient, appId);
    migrationClient.close();
}

public static void describeApplicationState(MigrationHubClient migrationClient,
String appId) {
    try {
        DescribeApplicationStateRequest applicationStateRequest =
DescribeApplicationStateRequest.builder()
            .applicationId(appId)
            .build();

        DescribeApplicationStateResponse applicationStateResponse =
migrationClient
            .describeApplicationState(applicationStateRequest);
        System.out.println("The application status is " +
applicationStateResponse.applicationStatusAsString());

    } catch (MigrationHubException e) {
        System.out.println(e.getMessage());
        System.exit(1);
    }
}
}
```

- Per i dettagli sull'API, [DescribeApplicationState](#) consulta AWS SDK for Java 2.x API Reference.



## DescribeMigrationTask

Il seguente esempio di codice mostra come utilizzare `DescribeMigrationTask`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.migrationhub.MigrationHubClient;
import
    software.amazon.awssdk.services.migrationhub.model.DescribeMigrationTaskRequest;
import
    software.amazon.awssdk.services.migrationhub.model.DescribeMigrationTaskResponse;
import software.amazon.awssdk.services.migrationhub.model.MigrationHubException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DescribeMigrationTask {

    public static void main(String[] args) {
        final String usage = ""

            Usage:
                DescribeMigrationTask <migrationTask> <progressStream>\s

            Where:
                migrationTask - the name of a migration task.\s
                progressStream - the name of a progress stream.\s

            """;

        if (args.length < 2) {
            System.out.println(usage);
        }
    }
}
```

```

        System.exit(1);
    }

    String migrationTask = args[0];
    String progressStream = args[1];
    Region region = Region.US_WEST_2;
    MigrationHubClient migrationClient = MigrationHubClient.builder()
        .region(region)
        .build();

    describeMigTask(migrationClient, migrationTask, progressStream);
    migrationClient.close();
}

public static void describeMigTask(MigrationHubClient migrationClient, String
migrationTask,
    String progressStream) {
    try {
        DescribeMigrationTaskRequest migrationTaskRequestRequest =
DescribeMigrationTaskRequest.builder()
            .progressUpdateStream(progressStream)
            .migrationTaskName(migrationTask)
            .build();

        DescribeMigrationTaskResponse migrationTaskResponse = migrationClient
            .describeMigrationTask(migrationTaskRequestRequest);
        System.out.println("The name is " +
migrationTaskResponse.migrationTask().migrationTaskName());

    } catch (MigrationHubException e) {
        System.out.println(e.getMessage());
        System.exit(1);
    }
}
}
}

```

- Per i dettagli sull'API, [DescribeMigrationTask](#) consulta AWS SDK for Java 2.x API Reference.

## ImportMigrationTask

Il seguente esempio di codice mostra come utilizzare `ImportMigrationTask`.

## SDK per Java 2.x

 Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.migrationhub.MigrationHubClient;
import
    software.amazon.awssdk.services.migrationhub.model.CreateProgressUpdateStreamRequest;
import
    software.amazon.awssdk.services.migrationhub.model.ImportMigrationTaskRequest;
import software.amazon.awssdk.services.migrationhub.model.MigrationHubException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ImportMigrationTask {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <migrationTask> <progressStream>\s

                Where:
                migrationTask - the name of a migration task.\s
                progressStream - the name of a progress stream.\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String migrationTask = args[0];
```

```

String progressStream = args[1];
Region region = Region.US_WEST_2;
MigrationHubClient migrationClient = MigrationHubClient.builder()
    .region(region)
    .build();

importMigrTask(migrationClient, migrationTask, progressStream);
migrationClient.close();
}

public static void importMigrTask(MigrationHubClient migrationClient, String
migrationTask, String progressStream) {
    try {
        CreateProgressUpdateStreamRequest progressUpdateStreamRequest =
CreateProgressUpdateStreamRequest.builder()
            .progressUpdateStreamName(progressStream)
            .dryRun(false)
            .build();

        migrationClient.createProgressUpdateStream(progressUpdateStreamRequest);
        ImportMigrationTaskRequest migrationTaskRequest =
ImportMigrationTaskRequest.builder()
            .migrationTaskName(migrationTask)
            .progressUpdateStream(progressStream)
            .dryRun(false)
            .build();

        migrationClient.importMigrationTask(migrationTaskRequest);

    } catch (MigrationHubException e) {
        System.out.println(e.getMessage());
        System.exit(1);
    }
}
}

```

- Per i dettagli sull'API, [ImportMigrationTask](#) consulta AWS SDK for Java 2.x API Reference.

## ListApplications

Il seguente esempio di codice mostra come utilizzare `ListApplications`.

## SDK per Java 2.x

 Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.migrationhub.MigrationHubClient;
import software.amazon.awssdk.services.migrationhub.model.ApplicationState;
import
    software.amazon.awssdk.services.migrationhub.model.ListApplicationStatesRequest;
import
    software.amazon.awssdk.services.migrationhub.model.ListApplicationStatesResponse;
import software.amazon.awssdk.services.migrationhub.model.MigrationHubException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListApplications {
    public static void main(String[] args) {
        Region region = Region.US_WEST_2;
        MigrationHubClient migrationClient = MigrationHubClient.builder()
            .region(region)
            .build();

        listApps(migrationClient);
        migrationClient.close();
    }

    public static void listApps(MigrationHubClient migrationClient) {
        try {
            ListApplicationStatesRequest applicationStatesRequest =
                ListApplicationStatesRequest.builder()
                    .maxResults(10)
```

```
        .build();

        ListApplicationStatesResponse response =
migrationClient.listApplicationStates(applicationStatesRequest);
        List<ApplicationState> apps = response.applicationStateList();
        for (ApplicationState appState : apps) {
            System.out.println("App Id is " + appState.applicationId());
            System.out.println("The status is " +
appState.applicationStatus().toString());
        }

    } catch (MigrationHubException e) {
        System.out.println(e.getMessage());
        System.exit(1);
    }
}
}
```

- Per i dettagli sull'API, [ListApplications](#) consulta AWS SDK for Java 2.x API Reference.

## ListCreatedArtifacts

Il seguente esempio di codice mostra come utilizzare `ListCreatedArtifacts`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.migrationhub.MigrationHubClient;
import software.amazon.awssdk.services.migrationhub.model.CreatedArtifact;
import
    software.amazon.awssdk.services.migrationhub.model.ListCreatedArtifactsRequest;
import
    software.amazon.awssdk.services.migrationhub.model.ListCreatedArtifactsResponse;
import software.amazon.awssdk.services.migrationhub.model.MigrationHubException;
import java.util.List;
```

```
/**
 * To run this Java V2 code example, ensure that you have setup your development
 * environment, including your credentials.
 *
 * For information, see this documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListCreatedArtifacts {
    public static void main(String[] args) {
        Region region = Region.US_WEST_2;
        MigrationHubClient migrationClient = MigrationHubClient.builder()
            .region(region)
            .build();

        listArtifacts(migrationClient);
        migrationClient.close();
    }

    public static void listArtifacts(MigrationHubClient migrationClient) {
        try {
            ListCreatedArtifactsRequest listCreatedArtifactsRequest =
            ListCreatedArtifactsRequest.builder()
                .maxResults(10)
                .migrationTaskName("SampleApp5")
                .progressUpdateStream("ProgressStreamB")
                .build();

            ListCreatedArtifactsResponse response =
            migrationClient.listCreatedArtifacts(listCreatedArtifactsRequest);
            List<CreatedArtifact> apps = response.createdArtifactList();
            for (CreatedArtifact artifact : apps) {
                System.out.println("App Id is " + artifact.description());
                System.out.println("The name is " + artifact.name());
            }

        } catch (MigrationHubException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Per i dettagli sull'API, [ListCreatedArtifacts](#) consulta AWS SDK for Java 2.x API Reference.

## ListMigrationTasks

Il seguente esempio di codice mostra come utilizzare `ListMigrationTasks`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.migrationhub.MigrationHubClient;
import software.amazon.awssdk.services.migrationhub.model.ListMigrationTasksRequest;
import
    software.amazon.awssdk.services.migrationhub.model.ListMigrationTasksResponse;
import software.amazon.awssdk.services.migrationhub.model.MigrationTaskSummary;
import software.amazon.awssdk.services.migrationhub.model.MigrationHubException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListMigrationTasks {
    public static void main(String[] args) {
        Region region = Region.US_WEST_2;
        MigrationHubClient migrationClient = MigrationHubClient.builder()
            .region(region)
            .build();

        listMigrTasks(migrationClient);
    }
}
```



```

        migrationClient.close();
    }

    public static void listMigrTasks(MigrationHubClient migrationClient) {
        try {
            ListMigrationTasksRequest listMigrationTasksRequest =
ListMigrationTasksRequest.builder()
                .maxResults(10)
                .build();

            ListMigrationTasksResponse response =
migrationClient.listMigrationTasks(listMigrationTasksRequest);
            List<MigrationTaskSummary> migrationList =
response.migrationTaskSummaryList();
            for (MigrationTaskSummary migration : migrationList) {
                System.out.println("Migration task name is " +
migration.migrationTaskName());
                System.out.println("The Progress update stream is " +
migration.progressUpdateStream());
            }

        } catch (MigrationHubException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}

```

- Per i dettagli sull'API, [ListMigrationTasks](#) consulta AWS SDK for Java 2.x API Reference.

## Esempi di Amazon Personalize con SDK for Java 2.x

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando Amazon Personalize. AWS SDK for Java 2.x

Le operazioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le operazioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati e negli esempi tra servizi.

Scenari: esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio.

Ogni esempio include un collegamento a GitHub, dove puoi trovare istruzioni su come configurare ed eseguire il codice nel contesto.

## Argomenti

- [Azioni](#)

## Azioni

### CreateBatchInferenceJob

Il seguente esempio di codice mostra come utilizzare `CreateBatchInferenceJob`.

SDK per Java 2.x

#### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public static String createPersonalizeBatchInferenceJob(PersonalizeClient
personalizeClient,
                String solutionVersionArn,
                String jobName,
                String s3InputDataSourcePath,
                String s3DataDestinationPath,
                String roleArn,
                String explorationWeight,
                String explorationItemAgeCutOff) {

    long waitInMilliseconds = 60 * 1000;
    String status;
    String batchInferenceJobArn;

    try {

        // Set up data input and output parameters.
        S3DataConfig inputSource = S3DataConfig.builder()
            .path(s3InputDataSourcePath)
            .build();
```

```

        S3DataConfig outputDestination = S3DataConfig.builder()
            .path(s3DataDestinationPath)
            .build();

        BatchInferenceJobInput jobInput =
BatchInferenceJobInput.builder()
            .s3DataSource(inputSource)
            .build();

        BatchInferenceJobOutput jobOutputLocation =
BatchInferenceJobOutput.builder()
            .s3DataDestination(outputDestination)
            .build();

        // Optional code to build the User-Personalization specific
item exploration
        // config.
        HashMap<String, String> explorationConfig = new HashMap<>();

        explorationConfig.put("explorationWeight",
explorationWeight);
        explorationConfig.put("explorationItemAgeCutOff",
explorationItemAgeCutOff);

        BatchInferenceJobConfig jobConfig =
BatchInferenceJobConfig.builder()
            .itemExplorationConfig(explorationConfig)
            .build();

        // End optional User-Personalization recipe specific code.

        CreateBatchInferenceJobRequest
createBatchInferenceJobRequest = CreateBatchInferenceJobRequest
            .builder()
            .solutionVersionArn(solutionVersionArn)
            .jobInput(jobInput)
            .jobOutput(jobOutputLocation)
            .jobName(jobName)
            .roleArn(roleArn)
            .batchInferenceJobConfig(jobConfig) //
Optional
            .build();

```

```

        batchInferenceJobArn =
personalizeClient.createBatchInferenceJob(createBatchInferenceJobRequest)
                    .batchInferenceJobArn();

        DescribeBatchInferenceJobRequest
describeBatchInferenceJobRequest = DescribeBatchInferenceJobRequest
                    .builder()
                    .batchInferenceJobArn(batchInferenceJobArn)
                    .build();

        long maxTime = Instant.now().getEpochSecond() + 3 * 60 * 60;
        while (Instant.now().getEpochSecond() < maxTime) {

                BatchInferenceJob batchInferenceJob =
personalizeClient

                .describeBatchInferenceJob(describeBatchInferenceJobRequest)
                    .batchInferenceJob();

                status = batchInferenceJob.status();
                System.out.println("Batch inference job status: " +
status);

                if (status.equals("ACTIVE") || status.equals("CREATE
FAILED")) {

                        break;
                }
                try {
                        Thread.sleep(waitInMilliseconds);
                } catch (InterruptedException e) {
                        System.out.println(e.getMessage());
                }
        }
        return batchInferenceJobArn;

    } catch (PersonalizeException e) {
        System.out.println(e.awsErrorDetails().errorMessage());
    }
    return "";
}

```

- Per i dettagli sull'API, [CreateBatchInferenceJob](#) consulta AWS SDK for Java 2.x API Reference.

## CreateCampaign

Il seguente esempio di codice mostra come utilizzare `CreateCampaign`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public static void createPersonalCampaign(PersonalizeClient personalizeClient,
String solutionVersionArn,
    String name) {

    try {
        CreateCampaignRequest createCampaignRequest =
CreateCampaignRequest.builder()
            .minProvisionedTPS(1)
            .solutionVersionArn(solutionVersionArn)
            .name(name)
            .build();

        CreateCampaignResponse campaignResponse =
personalizeClient.createCampaign(createCampaignRequest);
        System.out.println("The campaign ARN is " +
campaignResponse.campaignArn());

    } catch (PersonalizeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Per i dettagli sull'API, [CreateCampaign](#) consulta AWS SDK for Java 2.x API Reference.

## CreateDataset

Il seguente esempio di codice mostra come utilizzare `CreateDataset`.

## SDK per Java 2.x

 Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public static String createDataset(PersonalizeClient personalizeClient,
    String datasetName,
    String datasetGroupArn,
    String datasetType,
    String schemaArn) {
    try {
        CreateDatasetRequest request = CreateDatasetRequest.builder()
            .name(datasetName)
            .datasetGroupArn(datasetGroupArn)
            .datasetType(datasetType)
            .schemaArn(schemaArn)
            .build();

        String datasetArn = personalizeClient.createDataset(request)
            .datasetArn();
        System.out.println("Dataset " + datasetName + " created.");
        return datasetArn;

    } catch (PersonalizeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
```

- Per i dettagli sull'API, [CreateDataset](#) consulta AWS SDK for Java 2.x API Reference.

## CreateDatasetExportJob

Il seguente esempio di codice mostra come utilizzare `CreateDatasetExportJob`.

## SDK per Java 2.x

 Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public static String createDatasetExportJob(PersonalizeClient personalizeClient,
    String jobName,
    String datasetArn,
    IngestionMode ingestionMode,
    String roleArn,
    String s3BucketPath,
    String kmsKeyArn) {

    long waitInMilliseconds = 30 * 1000; // 30 seconds
    String status = null;

    try {

        S3DataConfig exportS3DataConfig =
        S3DataConfig.builder().path(s3BucketPath).kmsKeyArn(kmsKeyArn).build();
        DatasetExportJobOutput jobOutput =
        DatasetExportJobOutput.builder().s3DataDestination(exportS3DataConfig)
            .build();

        CreateDatasetExportJobRequest createRequest =
        CreateDatasetExportJobRequest.builder()
            .jobName(jobName)
            .datasetArn(datasetArn)
            .ingestionMode(ingestionMode)
            .jobOutput(jobOutput)
            .roleArn(roleArn)
            .build();

        String datasetExportJobArn =
        personalizeClient.createDatasetExportJob(createRequest).datasetExportJobArn();

        DescribeDatasetExportJobRequest describeDatasetExportJobRequest =
        DescribeDatasetExportJobRequest.builder()
            .datasetExportJobArn(datasetExportJobArn)
```

```
        .build();

    long maxTime = Instant.now().getEpochSecond() + 3 * 60 * 60;

    while (Instant.now().getEpochSecond() < maxTime) {

        DatasetExportJob datasetExportJob = personalizeClient
            .describeDatasetExportJob(describeDatasetExportJobRequest)
            .datasetExportJob();

        status = datasetExportJob.status();
        System.out.println("Export job status: " + status);

        if (status.equals("ACTIVE") || status.equals("CREATE FAILED")) {
            return status;
        }
        try {
            Thread.sleep(waitInMilliseconds);
        } catch (InterruptedException e) {
            System.out.println(e.getMessage());
        }
    }
} catch (PersonalizeException e) {
    System.out.println(e.awsErrorDetails().errorMessage());
}
return "";
}
```

- Per i dettagli sull'API, [CreateDatasetExportJob](#) consulta AWS SDK for Java 2.x API Reference.

## CreateDatasetGroup

Il seguente esempio di codice mostra come utilizzare `CreateDatasetGroup`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).



```
public static String createDatasetGroup(PersonalizeClient personalizeClient,
String datasetGroupName) {

    try {
        CreateDatasetGroupRequest createDatasetGroupRequest =
CreateDatasetGroupRequest.builder()
            .name(datasetGroupName)
            .build();

        return
personalizeClient.createDatasetGroup(createDatasetGroupRequest).datasetGroupArn();
    } catch (PersonalizeException e) {
        System.out.println(e.awsErrorDetails().errorMessage());
    }
    return "";
}
```

Crea un gruppo di set di dati di dominio.

```
public static String createDomainDatasetGroup(PersonalizeClient
personalizeClient,
        String datasetGroupName,
        String domain) {

    try {
        CreateDatasetGroupRequest createDatasetGroupRequest =
CreateDatasetGroupRequest.builder()
            .name(datasetGroupName)
            .domain(domain)
            .build();

        return
personalizeClient.createDatasetGroup(createDatasetGroupRequest).datasetGroupArn();
    } catch (PersonalizeException e) {
        System.out.println(e.awsErrorDetails().errorMessage());
    }
    return "";
}
```

- Per i dettagli sull'API, consulta la sezione [CreateDatasetGroup AWS SDK for Java 2.xAPI Reference](#).

## CreateDatasetImportJob

Il seguente esempio di codice mostra come utilizzare `CreateDatasetImportJob`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public static String createPersonalizeDatasetImportJob(PersonalizeClient
personalizeClient,
    String jobName,
    String datasetArn,
    String s3BucketPath,
    String roleArn) {

    long waitInMilliseconds = 60 * 1000;
    String status;
    String datasetImportJobArn;

    try {
        DataSource importDataSource = DataSource.builder()
            .dataLocation(s3BucketPath)
            .build();

        CreateDatasetImportJobRequest createDatasetImportJobRequest =
CreateDatasetImportJobRequest.builder()
            .datasetArn(datasetArn)
            .dataSource(importDataSource)
            .jobName(jobName)
            .roleArn(roleArn)
            .build();

        datasetImportJobArn =
personalizeClient.createDatasetImportJob(createDatasetImportJobRequest)
            .datasetImportJobArn();

        DescribeDatasetImportJobRequest describeDatasetImportJobRequest =
DescribeDatasetImportJobRequest.builder()
            .datasetImportJobArn(datasetImportJobArn)
            .build();
```

```
long maxTime = Instant.now().getEpochSecond() + 3 * 60 * 60;

while (Instant.now().getEpochSecond() < maxTime) {

    DatasetImportJob datasetImportJob = personalizeClient
        .describeDatasetImportJob(describeDatasetImportJobRequest)
        .datasetImportJob();

    status = datasetImportJob.status();
    System.out.println("Dataset import job status: " + status);

    if (status.equals("ACTIVE") || status.equals("CREATE FAILED")) {
        break;
    }
    try {
        Thread.sleep(waitInMilliseconds);
    } catch (InterruptedException e) {
        System.out.println(e.getMessage());
    }
}
return datasetImportJobArn;

} catch (PersonalizeException e) {
    System.out.println(e.awsErrorDetails().errorMessage());
}
return "";
}
```

- Per i dettagli sull'API, [CreateDatasetImportJob](#) consulta AWS SDK for Java 2.x API Reference.

## CreateEventTracker

Il seguente esempio di codice mostra come utilizzare `CreateEventTracker`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public static String createEventTracker(PersonalizeClient personalizeClient,
String eventTrackerName,
    String datasetGroupArn) {

    String eventTrackerId = "";
    String eventTrackerArn;
    long maxTime = 3 * 60 * 60; // 3 hours
    long waitInMilliseconds = 20 * 1000; // 20 seconds
    String status;

    try {

        CreateEventTrackerRequest createEventTrackerRequest =
CreateEventTrackerRequest.builder()
            .name(eventTrackerName)
            .datasetGroupArn(datasetGroupArn)
            .build();

        CreateEventTrackerResponse createEventTrackerResponse =
personalizeClient
            .createEventTracker(createEventTrackerRequest);

        eventTrackerArn = createEventTrackerResponse.eventTrackerArn();
        eventTrackerId = createEventTrackerResponse.trackingId();
        System.out.println("Event tracker ARN: " + eventTrackerArn);
        System.out.println("Event tracker ID: " + eventTrackerId);

        maxTime = Instant.now().getEpochSecond() + maxTime;

        DescribeEventTrackerRequest describeRequest =
DescribeEventTrackerRequest.builder()
            .eventTrackerArn(eventTrackerArn)
            .build();

        while (Instant.now().getEpochSecond() < maxTime) {

            status =
personalizeClient.describeEventTracker(describeRequest).eventTracker().status();
            System.out.println("EventTracker status: " + status);

            if (status.equals("ACTIVE") || status.equals("CREATE FAILED")) {
                break;
            }
        }
    }
}
```

```
        try {
            Thread.sleep(waitInMilliseconds);
        } catch (InterruptedException e) {
            System.out.println(e.getMessage());
        }
    }
    return eventTrackerId;
} catch (PersonalizeException e) {
    System.out.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
return eventTrackerId;
}
```

- Per i dettagli sull'API, [CreateEventTracker](#) consulta AWS SDK for Java 2.x API Reference.

## CreateFilter

Il seguente esempio di codice mostra come utilizzare `CreateFilter`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public static String createFilter(PersonalizeClient personalizeClient,
    String filterName,
    String datasetGroupArn,
    String filterExpression) {
    try {
        CreateFilterRequest request = CreateFilterRequest.builder()
            .name(filterName)
            .datasetGroupArn(datasetGroupArn)
            .filterExpression(filterExpression)
            .build();

        return personalizeClient.createFilter(request).filterArn();
    }
}
```

```
    } catch (PersonalizeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
```

- Per i dettagli sull'API, [CreateFilter](#) consulta AWS SDK for Java 2.x API Reference.

## CreateRecommender

Il seguente esempio di codice mostra come utilizzare `CreateRecommender`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public static String createRecommender(PersonalizeClient personalizeClient,
    String name,
    String datasetGroupArn,
    String recipeArn) {

    long maxTime = 0;
    long waitInMilliseconds = 30 * 1000; // 30 seconds
    String recommenderStatus = "";

    try {
        CreateRecommenderRequest createRecommenderRequest =
        CreateRecommenderRequest.builder()
            .datasetGroupArn(datasetGroupArn)
            .name(name)
            .recipeArn(recipeArn)
            .build();

        CreateRecommenderResponse recommenderResponse = personalizeClient
            .createRecommender(createRecommenderRequest);
    }
```

```

String recommenderArn = recommenderResponse.recommenderArn();
System.out.println("The recommender ARN is " + recommenderArn);

DescribeRecommenderRequest describeRecommenderRequest =
DescribeRecommenderRequest.builder()
    .recommenderArn(recommenderArn)
    .build();

maxTime = Instant.now().getEpochSecond() + 3 * 60 * 60;

while (Instant.now().getEpochSecond() < maxTime) {

    recommenderStatus =
personalizeClient.describeRecommender(describeRecommenderRequest).recommender()
    .status();
    System.out.println("Recommender status: " + recommenderStatus);

    if (recommenderStatus.equals("ACTIVE") ||
recommenderStatus.equals("CREATE FAILED")) {
        break;
    }
    try {
        Thread.sleep(waitInMilliseconds);
    } catch (InterruptedException e) {
        System.out.println(e.getMessage());
    }
}
return recommenderArn;

} catch (PersonalizeException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
return "";
}

```

- Per i dettagli sull'API, [CreateRecommender](#) consulta AWS SDK for Java 2.x API Reference.

## CreateSchema

Il seguente esempio di codice mostra come utilizzare CreateSchema.

## SDK per Java 2.x

 Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public static String createSchema(PersonalizeClient personalizeClient, String
schemaName, String filePath) {

    String schema = null;
    try {
        schema = new String(Files.readAllBytes(Paths.get(filePath)));
    } catch (IOException e) {
        System.out.println(e.getMessage());
    }

    try {
        CreateSchemaRequest createSchemaRequest = CreateSchemaRequest.builder()
            .name(schemaName)
            .schema(schema)
            .build();

        String schemaArn =
personalizeClient.createSchema(createSchemaRequest).schemaArn();

        System.out.println("Schema arn: " + schemaArn);

        return schemaArn;

    } catch (PersonalizeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
```

Crea uno schema con un dominio.



```
public static String createDomainSchema(PersonalizeClient personalizeClient,
String schemaName, String domain,
String filePath) {

String schema = null;
try {
    schema = new String(Files.readAllBytes(Paths.get(filePath)));
} catch (IOException e) {
    System.out.println(e.getMessage());
}

try {
    CreateSchemaRequest createSchemaRequest = CreateSchemaRequest.builder()
        .name(schemaName)
        .domain(domain)
        .schema(schema)
        .build();

String schemaArn =
personalizeClient.createSchema(createSchemaRequest).schemaArn();

    System.out.println("Schema arn: " + schemaArn);

    return schemaArn;


} catch (PersonalizeException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
return "";
}
```

- Per i dettagli sull'API, consulta la [CreateSchema](#) sezione AWS SDK for Java 2.x API Reference.

## CreateSolution

Il seguente esempio di codice mostra come utilizzare `CreateSolution`.

## SDK per Java 2.x

 Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public static String createPersonalizeSolution(PersonalizeClient
personalizeClient,
        String datasetGroupArn,
        String solutionName,
        String recipeArn) {

    try {
        CreateSolutionRequest solutionRequest = CreateSolutionRequest.builder()
            .name(solutionName)
            .datasetGroupArn(datasetGroupArn)
            .recipeArn(recipeArn)
            .build();

        CreateSolutionResponse solutionResponse =
personalizeClient.createSolution(solutionRequest);
        return solutionResponse.solutionArn();

    } catch (PersonalizeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
```

- Per i dettagli sull'API, [CreateSolution](#) consulta AWS SDK for Java 2.x API Reference.

## CreateSolutionVersion

Il seguente esempio di codice mostra come utilizzare `CreateSolutionVersion`.

## SDK per Java 2.x

 Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public static String createPersonalizeSolutionVersion(PersonalizeClient
personalizeClient, String solutionArn) {
    long maxTime = 0;
    long waitInMilliseconds = 30 * 1000; // 30 seconds
    String solutionStatus = "";
    String solutionVersionStatus = "";
    String solutionVersionArn = "";

    try {
        DescribeSolutionRequest describeSolutionRequest =
DescribeSolutionRequest.builder()
            .solutionArn(solutionArn)
            .build();

        maxTime = Instant.now().getEpochSecond() + 3 * 60 * 60;

        // Wait until solution is active.
        while (Instant.now().getEpochSecond() < maxTime) {

            solutionStatus =
personalizeClient.describeSolution(describeSolutionRequest).solution().status();
            System.out.println("Solution status: " + solutionStatus);

            if (solutionStatus.equals("ACTIVE") || solutionStatus.equals("CREATE
FAILED")) {
                break;
            }
            try {
                Thread.sleep(waitInMilliseconds);
            } catch (InterruptedException e) {
                System.out.println(e.getMessage());
            }
        }
    }
}
```

```
        if (solutionStatus.equals("ACTIVE")) {

            CreateSolutionVersionRequest createSolutionVersionRequest =
CreateSolutionVersionRequest.builder()
                .solutionArn(solutionArn)
                .build();

            CreateSolutionVersionResponse createSolutionVersionResponse =
personalizeClient
                .createSolutionVersion(createSolutionVersionRequest);
            solutionVersionArn =
createSolutionVersionResponse.solutionVersionArn();

            System.out.println("Solution version ARN: " + solutionVersionArn);

            DescribeSolutionVersionRequest describeSolutionVersionRequest =
DescribeSolutionVersionRequest.builder()
                .solutionVersionArn(solutionVersionArn)
                .build();

            while (Instant.now().getEpochSecond() < maxTime) {

                solutionVersionStatus =
personalizeClient.describeSolutionVersion(describeSolutionVersionRequest)
                    .solutionVersion().status();
                System.out.println("Solution version status: " +
solutionVersionStatus);

                if (solutionVersionStatus.equals("ACTIVE") ||
solutionVersionStatus.equals("CREATE FAILED")) {
                    break;
                }
                try {
                    Thread.sleep(waitInMilliseconds);
                } catch (InterruptedException e) {
                    System.out.println(e.getMessage());
                }
            }
            return solutionVersionArn;
        }
    } catch (PersonalizeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

```
        return "";  
    }
```

- Per i dettagli sull'API, [CreateSolutionVersion](#) consulta AWS SDK for Java 2.x API Reference.

## DeleteCampaign

Il seguente esempio di codice mostra come utilizzare `DeleteCampaign`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).


```
public static void deleteSpecificCampaign(PersonalizeClient personalizeClient,  
String campaignArn) {  
  
    try {  
        DeleteCampaignRequest campaignRequest = DeleteCampaignRequest.builder()  
            .campaignArn(campaignArn)  
            .build();  
  
        personalizeClient.deleteCampaign(campaignRequest);  
  
    } catch (PersonalizeException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

- Per i dettagli sull'API, [DeleteCampaign](#) consulta AWS SDK for Java 2.x API Reference.

## DeleteEventTracker

Il seguente esempio di codice mostra come utilizzare `DeleteEventTracker`.

## SDK per Java 2.x

 Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public static void deleteEventTracker(PersonalizeClient personalizeClient,
String eventTrackerArn) {
    try {
        DeleteEventTrackerRequest deleteEventTrackerRequest =
DeleteEventTrackerRequest.builder()
            .eventTrackerArn(eventTrackerArn)
            .build();

        int status =
personalizeClient.deleteEventTracker(deleteEventTrackerRequest).sdkHttpResponse().statusCode();

        System.out.println("Status code:" + status);

    } catch (PersonalizeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Per i dettagli sull'API, consulta la [DeleteEventTracker](#) sezione AWS SDK for Java 2.x API Reference.

## DeleteSolution

Il seguente esempio di codice mostra come utilizzare DeleteSolution.

## SDK per Java 2.x

 Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public static void deleteGivenSolution(PersonalizeClient personalizeClient,
String solutionArn) {

    try {
        DeleteSolutionRequest solutionRequest = DeleteSolutionRequest.builder()
            .solutionArn(solutionArn)
            .build();

        personalizeClient.deleteSolution(solutionRequest);
        System.out.println("Done");


    } catch (PersonalizeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Per i dettagli sull'API, consulta la [DeleteSolution](#) sezione AWS SDK for Java 2.x API Reference.

## DescribeCampaign

Il seguente esempio di codice mostra come utilizzare `DescribeCampaign`.

## SDK per Java 2.x

 Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public static void describeSpecificCampaign(PersonalizeClient personalizeClient,
String campaignArn) {

    try {
        DescribeCampaignRequest campaignRequest =
DescribeCampaignRequest.builder()
            .campaignArn(campaignArn)
            .build();

        DescribeCampaignResponse campaignResponse =
personalizeClient.describeCampaign(campaignRequest);
        Campaign myCampaign = campaignResponse.campaign();
        System.out.println("The Campaign name is " + myCampaign.name());
        System.out.println("The Campaign status is " + myCampaign.status());

    } catch (PersonalizeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Per i dettagli sull'API, consulta la [DescribeCampaign](#) sezione AWS SDK for Java 2.x API Reference.

## DescribeRecipe

Il seguente esempio di codice mostra come utilizzare `DescribeRecipe`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public static void describeSpecificRecipe(PersonalizeClient personalizeClient,
String recipeArn) {

    try {
```



```
DescribeRecipeRequest recipeRequest = DescribeRecipeRequest.builder()
    .recipeArn(recipeArn)
    .build();

DescribeRecipeResponse recipeResponse =
personalizeClient.describeRecipe(recipeRequest);
System.out.println("The recipe name is " +
recipeResponse.recipe().name());

    } catch (PersonalizeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Per i dettagli sull'API, consulta la [DescribeRecipe](#) sezione AWS SDK for Java 2.x API Reference.

## DescribeSolution

Il seguente esempio di codice mostra come utilizzare `DescribeSolution`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public static void describeSpecificSolution(PersonalizeClient personalizeClient,
String solutionArn) {

    try {
        DescribeSolutionRequest solutionRequest =
DescribeSolutionRequest.builder()
            .solutionArn(solutionArn)
            .build();
```

```
        DescribeSolutionResponse response =
personalizeClient.describeSolution(solutionRequest);
        System.out.println("The Solution name is " +
response.solution().name());

    } catch (PersonalizeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Per i dettagli sull'API, consulta la [DescribeSolution](#) sezione AWS SDK for Java 2.x API Reference.

## ListCampaigns

Il seguente esempio di codice mostra come utilizzare `ListCampaigns`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public static void listAllCampaigns(PersonalizeClient personalizeClient, String
solutionArn) {

    try {
        ListCampaignsRequest campaignsRequest = ListCampaignsRequest.builder()
            .maxResults(10)
            .solutionArn(solutionArn)
            .build();

        ListCampaignsResponse response =
personalizeClient.listCampaigns(campaignsRequest);
        List<CampaignSummary> campaigns = response.campaigns();
        for (CampaignSummary campaign : campaigns) {
            System.out.println("Campaign name is : " + campaign.name());
        }
    }
}
```

```
        System.out.println("Campaign ARN is : " + campaign.campaignArn());
    }

    } catch (PersonalizeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Per i dettagli sull'API, consulta la [ListCampaigns](#) sezione AWS SDK for Java 2.x API Reference.

## ListDatasetGroups

Il seguente esempio di codice mostra come utilizzare `ListDatasetGroups`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public static void listDSGroups(PersonalizeClient personalizeClient) {

    try {
        ListDatasetGroupsRequest groupsRequest =
ListDatasetGroupsRequest.builder()
            .maxResults(15)
            .build();

        ListDatasetGroupsResponse groupsResponse =
personalizeClient.listDatasetGroups(groupsRequest);
        List<DatasetGroupSummary> groups = groupsResponse.datasetGroups();
        for (DatasetGroupSummary group : groups) {
            System.out.println("The DataSet name is : " + group.name());
            System.out.println("The DataSet ARN is : " +
group.datasetGroupArn());
        }
    }
}
```

```
    } catch (PersonalizeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Per i dettagli sull'API, consulta la [ListDatasetGroups](#) sezione AWS SDK for Java 2.x API Reference.

## ListRecipes

Il seguente esempio di codice mostra come utilizzare `ListRecipes`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public static void listAllRecipes(PersonalizeClient personalizeClient) {

    try {
        ListRecipesRequest recipesRequest = ListRecipesRequest.builder()
            .maxResults(15)
            .build();

        ListRecipesResponse response =
personalizeClient.listRecipes(recipesRequest);
        List<RecipeSummary> recipes = response.recipes();
        for (RecipeSummary recipe : recipes) {
            System.out.println("The recipe ARN is: " + recipe.recipeArn());
            System.out.println("The recipe name is: " + recipe.name());
        }

    } catch (PersonalizeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

```
}
```

- Per i dettagli sull'API, consulta la [ListRecipessezione AWS SDK for Java 2.x API Reference](#).

## ListSolutions

Il seguente esempio di codice mostra come utilizzare `ListSolutions`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public static void listAllSolutions(PersonalizeClient personalizeClient, String
datasetGroupArn) {

    try {
        ListSolutionsRequest solutionsRequest = ListSolutionsRequest.builder()
            .maxResults(10)
            .datasetGroupArn(datasetGroupArn)
            .build();

        ListSolutionsResponse response =
personalizeClient.listSolutions(solutionsRequest);
        List<SolutionSummary> solutions = response.solutions();
        for (SolutionSummary solution : solutions) {
            System.out.println("The solution ARN is: " +
solution.solutionArn());
            System.out.println("The solution name is: " + solution.name());
        }

    } catch (PersonalizeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Per i dettagli sull'API, consulta la [ListSolutions](#) sezione AWS SDK for Java 2.x API Reference.

## UpdateCampaign

Il seguente esempio di codice mostra come utilizzare `UpdateCampaign`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public static String updateCampaign(PersonalizeClient personalizeClient,
    String campaignArn,
    String solutionVersionArn,
    Integer minProvisionedTPS) {

    try {
        // build the updateCampaignRequest
        UpdateCampaignRequest updateCampaignRequest =
UpdateCampaignRequest.builder()
            .campaignArn(campaignArn)
            .solutionVersionArn(solutionVersionArn)
            .minProvisionedTPS(minProvisionedTPS)
            .build();

        // update the campaign
        personalizeClient.updateCampaign(updateCampaignRequest);

        DescribeCampaignRequest campaignRequest =
DescribeCampaignRequest.builder()
            .campaignArn(campaignArn)
            .build();

        DescribeCampaignResponse campaignResponse =
personalizeClient.describeCampaign(campaignRequest);
        Campaign updatedCampaign = campaignResponse.campaign();

        System.out.println("The Campaign status is " +
updatedCampaign.status());
    }
}
```

```
        return updatedCampaign.status();

    } catch (PersonalizeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
```

- Per i dettagli sull'API, consulta la [UpdateCampaign](#) sezione AWS SDK for Java 2.x API Reference.

## Esempi di Amazon Personalize Events con SDK for Java 2.x

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando Amazon Personalize Events. AWS SDK for Java 2.x

Le operazioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le operazioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati e negli esempi tra servizi.

Scenari: esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio.

Ogni esempio include un collegamento a GitHub, dove puoi trovare istruzioni su come configurare ed eseguire il codice nel contesto.

### Argomenti

- [Azioni](#)

### Azioni

#### **PutEvents**

Il seguente esempio di codice mostra come utilizzare `PutEvents`.

## SDK per Java 2.x

 Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public static int putItems(PersonalizeEventsClient personalizeEventsClient,
    String datasetArn,
    String item1Id,
    String item1PropertyName,
    String item1PropertyValue,
    String item2Id,
    String item2PropertyName,
    String item2PropertyValue) {

    int responseCode = 0;
    ArrayList<Item> items = new ArrayList<>();

    try {
        Item item1 = Item.builder()
            .itemId(item1Id)
            .properties(String.format("{\ \"%1$s\ ": \"%2$s
\}"),
                item1PropertyName,
                item1PropertyValue))
            .build();

        items.add(item1);

        Item item2 = Item.builder()
            .itemId(item2Id)
            .properties(String.format("{\ \"%1$s\ ": \"%2$s
\}"),
                item2PropertyName,
                item2PropertyValue))
            .build();

        items.add(item2);

        PutItemsRequest putItemsRequest = PutItemsRequest.builder()
```



```

        .datasetArn(datasetArn)
        .items(items)
        .build();

        responseCode =
personalizeEventsClient.putItems(putItemsRequest).sdkHttpResponse().statusCode();
        System.out.println("Response code: " + responseCode);
        return responseCode;

    } catch (PersonalizeEventsException e) {
        System.out.println(e.awsErrorDetails().errorMessage());
    }
    return responseCode;
}

```

- Per i dettagli sull'API, consulta la [PutEvents](#) sezione AWS SDK for Java 2.x API Reference.

## PutUsers

Il seguente esempio di codice mostra come utilizzare `PutUsers`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```

public static int putUsers(PersonalizeEventsClient personalizeEventsClient,
    String datasetArn,
    String user1Id,
    String user1PropertyName,
    String user1PropertyValue,
    String user2Id,
    String user2PropertyName,
    String user2PropertyValue) {

    int responseCode = 0;
    ArrayList<User> users = new ArrayList<>();

```

```
        try {
            User user1 = User.builder()
                .userId(user1Id)
                .properties(String.format("{\\\"%1$s\\\": \\\"%2$s
\\}\",
                                user1PropertyName,
                                user1PropertyValue))
                .build();

            users.add(user1);

            User user2 = User.builder()
                .userId(user2Id)
                .properties(String.format("{\\\"%1$s\\\": \\\"%2$s
\\}\",
                                user2PropertyName,
                                user2PropertyValue))
                .build();

            users.add(user2);

            PutUsersRequest putUsersRequest = PutUsersRequest.builder()
                .datasetArn(datasetArn)
                .users(users)
                .build();

            responseCode =
personalizeEventsClient.putUsers(putUsersRequest).sdkHttpResponse().statusCode();
            System.out.println("Response code: " + responseCode);
            return responseCode;

        } catch (PersonalizeEventsException e) {
            System.out.println(e.awsErrorDetails().errorMessage());
        }
        return responseCode;
    }
}
```

- Per i dettagli sull'API, consulta la [PutUsers](#) sezione AWS SDK for Java 2.x API Reference.

## Esempi di Amazon Personalize Runtime con SDK for Java 2.x

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando AWS SDK for Java 2.x with Amazon Personalize Runtime.

Le operazioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le operazioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati e negli esempi tra servizi.

Scenari: esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio.

Ogni esempio include un collegamento a GitHub, dove puoi trovare istruzioni su come configurare ed eseguire il codice nel contesto.

### Argomenti

- [Azioni](#)

### Azioni

#### **GetPersonalizedRanking**

Il seguente esempio di codice mostra come utilizzare `GetPersonalizedRanking`.

SDK per Java 2.x

#### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public static List<PredictedItem> getRankedRecs(PersonalizeRuntimeClient
personalizeRuntimeClient,
        String campaignArn,
        String userId,
        ArrayList<String> items) {

    try {
        GetPersonalizedRankingRequest rankingRecommendationsRequest =
        GetPersonalizedRankingRequest.builder()
```

```

        .campaignArn(campaignArn)
        .userId(userId)
        .inputList(items)
        .build();

    GetPersonalizedRankingResponse recommendationsResponse =
personalizeRuntimeClient
        .getPersonalizedRanking(rankingRecommendationsRequest);
    List<PredictedItem> rankedItems =
recommendationsResponse.personalizedRanking();
    int rank = 1;
    for (PredictedItem item : rankedItems) {
        System.out.println("Item ranked at position " + rank + " details");
        System.out.println("Item Id is : " + item.itemId());
        System.out.println("Item score is : " + item.score());
        System.out.println("-----");
        rank++;
    }
    return rankedItems;
} catch (PersonalizeRuntimeException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
return null;
}

```

- Per i dettagli sull'API, consulta la [GetPersonalizedRanking](#) sezione AWS SDK for Java 2.x API Reference.

## GetRecommendations

Il seguente esempio di codice mostra come utilizzare `GetRecommendations`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Ottieni un elenco di articoli consigliati.

```
public static void getRecs(PersonalizeRuntimeClient personalizeRuntimeClient,
String campaignArn, String userId) {

    try {
        GetRecommendationsRequest recommendationsRequest =
GetRecommendationsRequest.builder()
            .campaignArn(campaignArn)
            .numResults(20)
            .userId(userId)
            .build();

        GetRecommendationsResponse recommendationsResponse =
personalizeRuntimeClient
            .getRecommendations(recommendationsRequest);
        List<PredictedItem> items = recommendationsResponse.itemList();
        for (PredictedItem item : items) {
            System.out.println("Item Id is : " + item.itemId());
            System.out.println("Item score is : " + item.score());
        }

    } catch (AwsServiceException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

Ottieni un elenco di elementi consigliati da un programma di consulenza creato in un gruppo di set di dati di dominio.

```
public static void getRecs(PersonalizeRuntimeClient personalizeRuntimeClient,
String recommenderArn,
    String userId) {

    try {
        GetRecommendationsRequest recommendationsRequest =
GetRecommendationsRequest.builder()
            .recommenderArn(recommenderArn)
            .numResults(20)
            .userId(userId)
            .build();
```

```

        GetRecommendationsResponse recommendationsResponse =
personalizeRuntimeClient
            .getRecommendations(recommendationsRequest);
        List<PredictedItem> items = recommendationsResponse.itemList();

        for (PredictedItem item : items) {
            System.out.println("Item Id is : " + item.itemId());
            System.out.println("Item score is : " + item.score());
        }
    } catch (AwsServiceException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

```

Usa un filtro quando richiedi consigli.

```

public static void getFilteredRecs(PersonalizeRuntimeClient
personalizeRuntimeClient,
    String campaignArn,
    String userId,
    String filterArn,
    String parameter1Name,
    String parameter1Value1,
    String parameter1Value2,
    String parameter2Name,
    String parameter2Value) {

    try {

        Map<String, String> filterValues = new HashMap<>();

        filterValues.put(parameter1Name, String.format("\'%1$s\'",\'%2$s\'",
            parameter1Value1, parameter1Value2));
        filterValues.put(parameter2Name, String.format("\'%1$s\'",
            parameter2Value));

        GetRecommendationsRequest recommendationsRequest =
GetRecommendationsRequest.builder()
            .campaignArn(campaignArn)
            .numResults(20)

```

```
        .userId(userId)
        .filterArn(filterArn)
        .filterValues(filterValues)
        .build();

    GetRecommendationsResponse recommendationsResponse =
personalizeRuntimeClient
        .getRecommendations(recommendationsRequest);
    List<PredictedItem> items = recommendationsResponse.itemList();

    for (PredictedItem item : items) {
        System.out.println("Item Id is : " + item.itemId());
        System.out.println("Item score is : " + item.score());
    }
} catch (PersonalizeRuntimeException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

- Per i dettagli sull'API, consulta la sezione [GetRecommendations AWS SDK for Java 2.x API Reference](#).

## Esempi di Amazon Pinpoint con SDK for Java 2.x

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando Amazon Pinpoint. AWS SDK for Java 2.x

Le operazioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le operazioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati e negli esempi tra servizi.

Scenari: esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio.

Ogni esempio include un collegamento a GitHub, dove puoi trovare istruzioni su come configurare ed eseguire il codice nel contesto.

### Argomenti

- [Azioni](#)

## Azioni

### CreateApp

Il seguente esempio di codice mostra come utilizzare `CreateApp`.

SDK per Java 2.x

#### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.CreateAppRequest;
import software.amazon.awssdk.services.pinpoint.model.CreateAppResponse;
import software.amazon.awssdk.services.pinpoint.model.CreateApplicationRequest;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateApp {
    public static void main(String[] args) {
        final String usage = ""

            Usage: <appName>

            Where:
            appName - The name of the application to create.

            """;

        if (args.length != 1) {
            System.out.println(usage);
        }
    }
}
```



```

        System.exit(1);
    }
    String appName = args[0];
    System.out.println("Creating an application with name: " + appName);

    PinpointClient pinpoint = PinpointClient.builder()
        .region(Region.US_EAST_1)
        .build();

    String appID = createApplication(pinpoint, appName);
    System.out.println("App ID is: " + appID);
    pinpoint.close();
}

public static String createApplication(PinpointClient pinpoint, String appName)
{
    try {
        CreateApplicationRequest appRequest = CreateApplicationRequest.builder()
            .name(appName)
            .build();

        CreateAppRequest request = CreateAppRequest.builder()
            .createApplicationRequest(appRequest)
            .build();

        CreateAppResponse result = pinpoint.createApp(request);
        return result.applicationResponse().id();

    } catch (PinpointException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
}

```

- Per i dettagli sull'API, consulta la [CreateApp](#) sezione AWS SDK for Java 2.x API Reference.

## CreateCampaign

Il seguente esempio di codice mostra come utilizzare `CreateCampaign`.

## SDK per Java 2.x

 Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

## Crea una campagna.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.CampaignResponse;
import software.amazon.awssdk.services.pinpoint.model.Message;
import software.amazon.awssdk.services.pinpoint.model.Schedule;
import software.amazon.awssdk.services.pinpoint.model.Action;
import software.amazon.awssdk.services.pinpoint.model.MessageConfiguration;
import software.amazon.awssdk.services.pinpoint.model.WriteCampaignRequest;
import software.amazon.awssdk.services.pinpoint.model.CreateCampaignResponse;
import software.amazon.awssdk.services.pinpoint.model.CreateCampaignRequest;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateCampaign {
    public static void main(String[] args) {

        final String usage = ""

            Usage:  <appId> <segmentId>

            Where:
                appId - The ID of the application to create the campaign in.
                segmentId - The ID of the segment to create the campaign from.
            """;

        if (args.length != 2) {
```

```
        System.out.println(usage);
        System.exit(1);
    }

    String appId = args[0];
    String segmentId = args[1];
    PinpointClient pinpoint = PinpointClient.builder()
        .region(Region.US_EAST_1)
        .build();

    createPinCampaign(pinpoint, appId, segmentId);
    pinpoint.close();
}

public static void createPinCampaign(PinpointClient pinpoint, String appId,
String segmentId) {
    CampaignResponse result = createCampaign(pinpoint, appId, segmentId);
    System.out.println("Campaign " + result.name() + " created.");
    System.out.println(result.description());
}

public static CampaignResponse createCampaign(PinpointClient client, String
appId, String segmentID) {

    try {
        Schedule schedule = Schedule.builder()
            .startTime("IMMEDIATE")
            .build();

        Message defaultMessage = Message.builder()
            .action(Action.OPEN_APP)
            .body("My message body.")
            .title("My message title.")
            .build();

        MessageConfiguration messageConfiguration =
MessageConfiguration.builder()
            .defaultMessage(defaultMessage)
            .build();

        WriteCampaignRequest request = WriteCampaignRequest.builder()
            .description("My description")
            .schedule(schedule)
            .name("MyCampaign")
```

```
        .segmentId(segmentID)
        .messageConfiguration(messageConfiguration)
        .build();

        CreateCampaignResponse result =
client.createCampaign(CreateCampaignRequest.builder()
        .applicationId(appID)
        .writeCampaignRequest(request).build());

        System.out.println("Campaign ID: " + result.campaignResponse().id());
        return result.campaignResponse();

    } catch (PinpointException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }

    return null;
}
}
```

- Per i dettagli sull'API, consulta la [CreateCampaign](#) sezione AWS SDK for Java 2.x API Reference.

## CreateExportJob

Il seguente esempio di codice mostra come utilizzare `CreateExportJob`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Esporta un endpoint.

```
import software.amazon.awssdk.core.ResponseBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.PinpointClient;
```

```
import software.amazon.awssdk.services.pinpoint.model.ExportJobRequest;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;
import software.amazon.awssdk.services.pinpoint.model.CreateExportJobRequest;
import software.amazon.awssdk.services.pinpoint.model.CreateExportJobResponse;
import software.amazon.awssdk.services.pinpoint.model.GetExportJobResponse;
import software.amazon.awssdk.services.pinpoint.model.GetExportJobRequest;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Request;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Response;
import software.amazon.awssdk.services.s3.model.S3Object;
import software.amazon.awssdk.services.s3.model.GetObjectResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.OutputStream;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;
import java.util.List;
import java.util.concurrent.TimeUnit;
import java.util.stream.Collectors;

/**
 * To run this code example, you need to create an AWS Identity and Access
 * Management (IAM) role with the correct policy as described in this
 * documentation:
 * https://docs.aws.amazon.com/pinpoint/latest/developerguide/audience-data-export.html
 *
 * Also, set up your development environment, including your credentials.
 *
 * For information, see this documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class ExportEndpoints {
    public static void main(String[] args) {
        final String usage = ""
```

This program performs the following steps:

1. Exports the endpoints to an Amazon S3 bucket.  
 2. Downloads the exported endpoints files from Amazon S3.  
 3. Parses the endpoints files to obtain the endpoint IDs and prints them.

Usage: ExportEndpoints <applicationId> <s3BucketName>  
 <iamExportRoleArn> <path>

Where:

applicationId - The ID of the Amazon Pinpoint application that has the endpoint.

s3BucketName - The name of the Amazon S3 bucket to export the JSON file to.\s

iamExportRoleArn - The ARN of an IAM role that grants Amazon Pinpoint write permissions to the S3 bucket. path - The path where the files downloaded from the Amazon S3 bucket are written (for example, C:/AWS/).

```

    """;

    if (args.length != 4) {
        System.out.println(usage);
        System.exit(1);
    }

    String applicationId = args[0];
    String s3BucketName = args[1];
    String iamExportRoleArn = args[2];
    String path = args[3];
    System.out.println("Deleting an application with ID: " + applicationId);

    Region region = Region.US_EAST_1;
    PinpointClient pinpoint = PinpointClient.builder()
        .region(region)
        .build();

    S3Client s3Client = S3Client.builder()
        .region(region)
        .build();

    exportAllEndpoints(pinpoint, s3Client, applicationId, s3BucketName, path,
iamExportRoleArn);
    pinpoint.close();
    s3Client.close();
}

public static void exportAllEndpoints(PinpointClient pinpoint,

```

```

        S3Client s3Client,
        String applicationId,
        String s3BucketName,
        String path,
        String iamExportRoleArn) {

    try {
        List<String> objectKeys = exportEndpointsToS3(pinpoint, s3Client,
s3BucketName, iamExportRoleArn,
            applicationId);
        List<String> endpointFileKeys = objectKeys.stream().filter(o ->
o.endsWith(".gz"))
            .collect(Collectors.toList());
        downloadFromS3(s3Client, path, s3BucketName, endpointFileKeys);

    } catch (PinpointException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static List<String> exportEndpointsToS3(PinpointClient pinpoint, S3Client
s3Client, String s3BucketName,
        String iamExportRoleArn, String applicationId) {

    SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd-
HH_mm:ss.SSS_z");
    String endpointsKeyPrefix = "exports/" + applicationId + "_" +
dateFormat.format(new Date());
    String s3UrlPrefix = "s3://" + s3BucketName + "/" + endpointsKeyPrefix +
"/";
    List<String> objectKeys = new ArrayList<>();
    String key;

    try {
        // Defines the export job that Amazon Pinpoint runs.
        ExportJobRequest jobRequest = ExportJobRequest.builder()
            .roleArn(iamExportRoleArn)
            .s3UrlPrefix(s3UrlPrefix)
            .build();

        CreateExportJobRequest exportJobRequest =
CreateExportJobRequest.builder()
            .applicationId(applicationId)

```

```
        .exportJobRequest(jobRequest)
        .build();

        System.out.format("Exporting endpoints from Amazon Pinpoint application
%s to Amazon S3 " +
        "bucket %s . . .\n", applicationId, s3BucketName);

        CreateExportJobResponse exportResult =
pinpoint.createExportJob(exportJobRequest);
        String jobId = exportResult.exportJobResponse().id();
        System.out.println(jobId);
        printExportJobStatus(pinpoint, applicationId, jobId);

        ListObjectsV2Request v2Request = ListObjectsV2Request.builder()
            .bucket(s3BucketName)
            .prefix(endpointsKeyPrefix)
            .build();

        // Create a list of object keys.
        ListObjectsV2Response v2Response = s3Client.listObjectsV2(v2Request);
        List<S3Object> objects = v2Response.contents();
        for (S3Object object : objects) {
            key = object.key();
            objectKeys.add(key);
        }

        return objectKeys;

    } catch (PinpointException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return null;
}

private static void printExportJobStatus(PinpointClient pinpointClient,
        String applicationId,
        String jobId) {

    GetExportJobResponse getExportJobResult;
    String status;

    try {
        // Checks the job status until the job completes or fails.
```



```
GetExportJobRequest exportJobRequest = GetExportJobRequest.builder()
    .jobId(jobId)
    .applicationId(applicationId)
    .build();

do {
    getExportJobResult = pinpointClient.getExportJob(exportJobRequest);
    status =
getExportJobResult.exportJobResponse().jobStatus().toString().toUpperCase();
    System.out.format("Export job %s . . .\n", status);
    TimeUnit.SECONDS.sleep(3);

} while (!status.equals("COMPLETED") && !status.equals("FAILED"));

if (status.equals("COMPLETED")) {
    System.out.println("Finished exporting endpoints.");
} else {
    System.err.println("Failed to export endpoints.");
    System.exit(1);
}

} catch (PinpointException | InterruptedException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
}

// Download files from an Amazon S3 bucket and write them to the path location.
public static void downloadFromS3(S3Client s3Client, String path, String
s3BucketName, List<String> objectKeys) {

    String newPath;
    try {
        for (String key : objectKeys) {
            GetObjectRequest objectRequest = GetObjectRequest.builder()
                .bucket(s3BucketName)
                .key(key)
                .build();

            ResponseBytes<GetObjectResponse> objectBytes =
s3Client.getObjectAsBytes(objectRequest);
            byte[] data = objectBytes.asByteArray();

            // Write the data to a local file.
```

```
        String fileSuffix = new
SimpleDateFormat("yyyyMMddHHmmss").format(new Date());
        newPath = path + fileSuffix + ".gz";
        File myFile = new File(newPath);
        OutputStream os = new FileOutputStream(myFile);
        os.write(data);
    }
    System.out.println("Download finished.");

} catch (S3Exception | NullPointerException | IOException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
}
}
```

- Per i dettagli sull'API, consulta la [CreateExportJob](#) sezione AWS SDK for Java 2.x API Reference.

## CreateImportJob

Il seguente esempio di codice mostra come utilizzare `CreateImportJob`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Importa un segmento.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.CreateImportJobRequest;
import software.amazon.awssdk.services.pinpoint.model.ImportJobResponse;
import software.amazon.awssdk.services.pinpoint.model.ImportJobRequest;
import software.amazon.awssdk.services.pinpoint.model.Format;
import software.amazon.awssdk.services.pinpoint.model.CreateImportJobResponse;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;
```

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ImportSegment {
    public static void main(String[] args) {
        final String usage = ""

            Usage:  <appId> <bucket> <key> <roleArn>\s

            Where:
                appId - The application ID to create a segment for.
                bucket - The name of the Amazon S3 bucket that contains the
segment definitons.
                key - The key of the S3 object.
                roleArn - ARN of the role that allows Amazon Pinpoint to
access S3. You need to set trust management for this to work. See https://docs.aws.amazon.com/IAM/latest/UserGuide/reference\_policies\_elements\_principal.html
                """;

        if (args.length != 4) {
            System.out.println(usage);
            System.exit(1);
        }

        String appId = args[0];
        String bucket = args[1];
        String key = args[2];
        String roleArn = args[3];

        PinpointClient pinpoint = PinpointClient.builder()
            .region(Region.US_EAST_1)
            .build();

        ImportJobResponse response = createImportSegment(pinpoint, appId, bucket,
key, roleArn);
        System.out.println("Import job for " + bucket + " submitted.");
        System.out.println("See application " + response.applicationId() + " for
import job status.");
    }
}
```

```
        System.out.println("See application " + response.jobStatus() + " for import
job status.");
        pinpoint.close();
    }

    public static ImportJobResponse createImportSegment(PinpointClient client,
        String appId,
        String bucket,
        String key,
        String roleArn) {

        try {
            ImportJobRequest importRequest = ImportJobRequest.builder()
                .defineSegment(true)
                .registerEndpoints(true)
                .roleArn(roleArn)
                .format(Format.JSON)
                .s3Url("s3://" + bucket + "/" + key)
                .build();

            CreateImportJobRequest jobRequest = CreateImportJobRequest.builder()
                .importJobRequest(importRequest)
                .applicationId(appId)
                .build();

            CreateImportJobResponse jobResponse =
client.createImportJob(jobRequest);
            return jobResponse.importJobResponse();

        } catch (PinpointException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return null;
    }
}
```

- Per i dettagli sull'API, consulta la [CreateImportJob](#) sezione AWS SDK for Java 2.x API Reference.

## CreateSegment

Il seguente esempio di codice mostra come utilizzare `CreateSegment`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.AttributeDimension;
import software.amazon.awssdk.services.pinpoint.model.SegmentResponse;
import software.amazon.awssdk.services.pinpoint.model.AttributeType;
import software.amazon.awssdk.services.pinpoint.model.RecencyDimension;
import software.amazon.awssdk.services.pinpoint.model.SegmentBehaviors;
import software.amazon.awssdk.services.pinpoint.model.SegmentDemographics;
import software.amazon.awssdk.services.pinpoint.model.SegmentLocation;
import software.amazon.awssdk.services.pinpoint.model.SegmentDimensions;
import software.amazon.awssdk.services.pinpoint.model.WriteSegmentRequest;
import software.amazon.awssdk.services.pinpoint.model.CreateSegmentRequest;
import software.amazon.awssdk.services.pinpoint.model.CreateSegmentResponse;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;
import java.util.HashMap;
import java.util.Map;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateSegment {
    public static void main(String[] args) {
        final String usage = ""

                                Usage:  <appId>
```

```

        Where:
            appId - The application ID to create a segment
for.

        """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String appId = args[0];
        PinpointClient pinpoint = PinpointClient.builder()
            .region(Region.US_EAST_1)
            .build();

        SegmentResponse result = createSegment(pinpoint, appId);
        System.out.println("Segment " + result.name() + " created.");
        System.out.println(result.segmentType());
        pinpoint.close();
    }

    public static SegmentResponse createSegment(PinpointClient client, String
appId) {
        try {
            Map<String, AttributeDimension> segmentAttributes = new
HashMap<>();

            segmentAttributes.put("Team", AttributeDimension.builder()
                .attributeType(AttributeType.INCLUSIVE)
                .values("Lakers")
                .build());

            RecencyDimension recencyDimension =
RecencyDimension.builder()
                .duration("DAY_30")
                .recencyType("ACTIVE")
                .build();

            SegmentBehaviors segmentBehaviors =
SegmentBehaviors.builder()
                .recency(recencyDimension)
                .build();

```

```
SegmentDemographics segmentDemographics =
SegmentDemographics
    .builder()
    .build();

SegmentLocation segmentLocation = SegmentLocation
    .builder()
    .build();

SegmentDimensions dimensions = SegmentDimensions
    .builder()
    .attributes(segmentAttributes)
    .behavior(segmentBehaviors)
    .demographic(segmentDemographics)
    .location(segmentLocation)
    .build();

WriteSegmentRequest writeSegmentRequest =
WriteSegmentRequest.builder()
    .name("MySegment")
    .dimensions(dimensions)
    .build();

CreateSegmentRequest createSegmentRequest =
CreateSegmentRequest.builder()
    .applicationId(appId)
    .writeSegmentRequest(writeSegmentRequest)
    .build();

CreateSegmentResponse createSegmentResult =
client.createSegment(createSegmentRequest);
    System.out.println("Segment ID: " +
createSegmentResult.segmentResponse().id());
    System.out.println("Done");
    return createSegmentResult.segmentResponse();

    } catch (PinpointException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return null;
}
}
```

- Per i dettagli sull'API, consulta la [CreateSegment](#) sezione AWS SDK for Java 2.x API Reference.

## DeleteApp

Il seguente esempio di codice mostra come utilizzare `DeleteApp`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Elimina un'applicazione.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.DeleteAppRequest;
import software.amazon.awssdk.services.pinpoint.model.DeleteAppResponse;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteApp {
    public static void main(String[] args) {
        final String usage = ""

                Usage: <appId>

                Where:
                appId - The ID of the application to delete.
    }
}
```



```
        """);

    if (args.length != 1) {
        System.out.println(usage);
        System.exit(1);
    }

    String appId = args[0];
    System.out.println("Deleting an application with ID: " + appId);
    PinpointClient pinpoint = PinpointClient.builder()
        .region(Region.US_EAST_1)
        .build();

    deletePinApp(pinpoint, appId);
    System.out.println("Done");
    pinpoint.close();
}

public static void deletePinApp(PinpointClient pinpoint, String appId) {
    try {
        DeleteAppRequest appRequest = DeleteAppRequest.builder()
            .applicationId(appId)
            .build();

        DeleteAppResponse result = pinpoint.deleteApp(appRequest);
        String appName = result.applicationResponse().name();
        System.out.println("Application " + appName + " has been deleted.");

    } catch (PinpointException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Per i dettagli sull'API, consulta la [DeleteApp](#) sezione AWS SDK for Java 2.x API Reference.

## DeleteEndpoint

Il seguente esempio di codice mostra come utilizzare `DeleteEndpoint`.

## SDK per Java 2.x

 Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

## Eliminare un endpoint.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.DeleteEndpointRequest;
import software.amazon.awssdk.services.pinpoint.model.DeleteEndpointResponse;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteEndpoint {
    public static void main(String[] args) {
        final String usage = ""

            Usage:  <appName> <endpointId >

            Where:
                appId - The id of the application to delete.
                endpointId - The id of the endpoint to delete.
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String appId = args[0];
        String endpointId = args[1];
        System.out.println("Deleting an endpoint with id: " + endpointId);
```

```
PinpointClient pinpoint = PinpointClient.builder()
    .region(Region.US_EAST_1)
    .build();

deletePinEndpoint(pinpoint, appId, endpointId);
pinpoint.close();
}

public static void deletePinEndpoint(PinpointClient pinpoint, String appId,
String endpointId) {
    try {
        DeleteEndpointRequest appRequest = DeleteEndpointRequest.builder()
            .applicationId(appId)
            .endpointId(endpointId)
            .build();

        DeleteEndpointResponse result = pinpoint.deleteEndpoint(appRequest);
        String id = result.endpointResponse().id();
        System.out.println("The deleted endpoint id " + id);

    } catch (PinpointException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    System.out.println("Done");
}
}
```

- Per i dettagli sull'API, consulta la [DeleteEndpoint](#) sezione AWS SDK for Java 2.x API Reference.

## GetEndpoint

Il seguente esempio di codice mostra come utilizzare `GetEndpoint`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import com.google.gson.FieldNamingPolicy;
import com.google.gson.Gson;
import com.google.gson.GsonBuilder;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.EndpointResponse;
import software.amazon.awssdk.services.pinpoint.model.GetEndpointResponse;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;
import software.amazon.awssdk.services.pinpoint.model.GetEndpointRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class LookUpEndpoint {
    public static void main(String[] args) {
        final String usage = ""

            Usage:  <appId> <endpoint>

            Where:
                appId - The ID of the application to delete.
                endpoint - The ID of the endpoint.\s
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String appId = args[0];
        String endpoint = args[1];
        System.out.println("Looking up an endpoint point with ID: " + endpoint);
        PinpointClient pinpoint = PinpointClient.builder()
            .region(Region.US_EAST_1)
            .build();

        lookupPinpointEndpoint(pinpoint, appId, endpoint);
        pinpoint.close();
    }
}
```

```
    }

    public static void lookupPinpointEndpoint(PinpointClient pinpoint, String appId,
String endpoint) {
        try {
            GetEndpointRequest appRequest = GetEndpointRequest.builder()
                .applicationId(appId)
                .endpointId(endpoint)
                .build();

            GetEndpointResponse result = pinpoint.getEndpoint(appRequest);
            EndpointResponse endResponse = result.endpointResponse();

            // Uses the Google Gson library to pretty print the endpoint JSON.
            Gson gson = new GsonBuilder()
                .setFieldNamingPolicy(FieldNamingPolicy.UPPER_CAMEL_CASE)
                .setPrettyPrinting()
                .create();

            String endpointJson = gson.toJson(endResponse);
            System.out.println(endpointJson);

        } catch (PinpointException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        System.out.println("Done");
    }
}
```

- Per i dettagli sull'API, consulta la [GetEndpoint](#) sezione AWS SDK for Java 2.x API Reference.

## GetSegments

Il seguente esempio di codice mostra come utilizzare `GetSegments`.

## SDK per Java 2.x

 Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Elenca i segmenti.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.GetSegmentsRequest;
import software.amazon.awssdk.services.pinpoint.model.GetSegmentsResponse;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;
import software.amazon.awssdk.services.pinpoint.model.SegmentResponse;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListSegments {
    public static void main(String[] args) {
        final String usage = ""

                Usage:  <appId>

                Where:
                    appId - The ID of the application that contains a segment.

                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String appId = args[0];
```

```
PinpointClient pinpoint = PinpointClient.builder()
    .region(Region.US_EAST_1)
    .build();

listSegs(pinpoint, appId);
pinpoint.close();
}

public static void listSegs(PinpointClient pinpoint, String appId) {
    try {
        GetSegmentsRequest request = GetSegmentsRequest.builder()
            .applicationId(appId)
            .build();

        GetSegmentsResponse response = pinpoint.getSegments(request);
        List<SegmentResponse> segments = response.segmentsResponse().item();
        for (SegmentResponse segment : segments) {
            System.out
                .println("Segment " + segment.id() + " " + segment.name() +
                    " " + segment.lastModifiedDate());
        }

    } catch (PinpointException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Per i dettagli sull'API, consulta la [GetSegments](#) sezione AWS SDK for Java 2.x API Reference.

## GetSmsChannel

Il seguente esempio di codice mostra come utilizzare `GetSmsChannel`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.SMSChannelResponse;
import software.amazon.awssdk.services.pinpoint.model.GetSmsChannelRequest;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;
import software.amazon.awssdk.services.pinpoint.model.SMSChannelRequest;
import software.amazon.awssdk.services.pinpoint.model.UpdateSmsChannelRequest;
import software.amazon.awssdk.services.pinpoint.model.UpdateSmsChannelResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class UpdateChannel {
    public static void main(String[] args) {
        final String usage = ""

            Usage: CreateChannel <appId>

            Where:
                appId - The name of the application whose channel is updated.

            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String appId = args[0];
        PinpointClient pinpoint = PinpointClient.builder()
            .region(Region.US_EAST_1)
            .build();

        SMSChannelResponse getResponse = getSmsChannel(pinpoint, appId);
        toggleSmsChannel(pinpoint, appId, getResponse);
        pinpoint.close();
    }
}
```



```
private static SMSChannelResponse getSmsChannel(PinpointClient client, String
appId) {
    try {
        GetSmsChannelRequest request = GetSmsChannelRequest.builder()
            .applicationId(appId)
            .build();

        SMSChannelResponse response =
client.getSmsChannel(request).smsChannelResponse();
        System.out.println("Channel state is " + response.enabled());
        return response;

    } catch (PinpointException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return null;
}

private static void toggleSmsChannel(PinpointClient client, String appId,
SMSChannelResponse getResponse) {
    boolean enabled = !getResponse.enabled();
    try {
        SMSChannelRequest request = SMSChannelRequest.builder()
            .enabled(enabled)
            .build();

        UpdateSmsChannelRequest updateRequest =
UpdateSmsChannelRequest.builder()
            .smsChannelRequest(request)
            .applicationId(appId)
            .build();

        UpdateSmsChannelResponse result =
client.updateSmsChannel(updateRequest);
        System.out.println("Channel state: " +
result.smsChannelResponse().enabled());

    } catch (PinpointException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Per i dettagli sull'API, consulta la [GetSmsChannel](#) sezione AWS SDK for Java 2.x API Reference.

## GetUserEndpoints

Il seguente esempio di codice mostra come utilizzare `GetUserEndpoints`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.EndpointResponse;
import software.amazon.awssdk.services.pinpoint.model.GetUserEndpointsRequest;
import software.amazon.awssdk.services.pinpoint.model.GetUserEndpointsResponse;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListEndpointIds {
    public static void main(String[] args) {
        final String usage = ""

                Usage:    <applicationId> <userId>

                Where:
```

```
        applicationId - The ID of the Amazon Pinpoint application that
has the endpoint.
        userId - The user id applicable to the endpoints""";

    if (args.length != 2) {
        System.out.println(usage);
        System.exit(1);
    }

    String applicationId = args[0];
    String userId = args[1];
    PinpointClient pinpoint = PinpointClient.builder()
        .region(Region.US_EAST_1)
        .build();

    listAllEndpoints(pinpoint, applicationId, userId);
    pinpoint.close();
}

public static void listAllEndpoints(PinpointClient pinpoint,
    String applicationId,
    String userId) {

    try {
        GetUserEndpointsRequest endpointsRequest =
        GetUserEndpointsRequest.builder()
            .userId(userId)
            .applicationId(applicationId)
            .build();

        GetUserEndpointsResponse response =
        pinpoint.getUserEndpoints(endpointsRequest);
        List<EndpointResponse> endpoints = response.endpointsResponse().item();

        // Display the results.
        for (EndpointResponse endpoint : endpoints) {
            System.out.println("The channel type is: " +
            endpoint.channelType());
            System.out.println("The address is " + endpoint.address());
        }

    } catch (PinpointException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

```
    }  
  }  
}
```

- Per i dettagli sull'API, consulta la [GetUserEndpoints](#) sezione AWS SDK for Java 2.x API Reference.

## SendMessages

Il seguente esempio di codice mostra come utilizzare `SendMessages`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Invia un messaggio e-mail.

```
import software.amazon.awssdk.regions.Region;  
import software.amazon.awssdk.services.pinpoint.PinpointClient;  
import software.amazon.awssdk.services.pinpoint.model.AddressConfiguration;  
import software.amazon.awssdk.services.pinpoint.model.ChannelType;  
import software.amazon.awssdk.services.pinpoint.model.SimpleEmailPart;  
import software.amazon.awssdk.services.pinpoint.model.SimpleEmail;  
import software.amazon.awssdk.services.pinpoint.model.EmailMessage;  
import software.amazon.awssdk.services.pinpoint.model.DirectMessageConfiguration;  
import software.amazon.awssdk.services.pinpoint.model.MessageRequest;  
import software.amazon.awssdk.services.pinpoint.model.SendMessagesRequest;  
import software.amazon.awssdk.services.pinpoint.model.PinpointException;  
import software.amazon.awssdk.services.pinpointemail.PinpointEmailClient;  
import software.amazon.awssdk.services.pinpointemail.model.Body;  
import software.amazon.awssdk.services.pinpointemail.model.Content;  
import software.amazon.awssdk.services.pinpointemail.model.Destination;  
import software.amazon.awssdk.services.pinpointemail.model.EmailContent;  
import software.amazon.awssdk.services.pinpointemail.model.Message;  
import software.amazon.awssdk.services.pinpointemail.model.SendEmailRequest;  
  
import java.util.HashMap;
```

```

import java.util.Map;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class SendMessage {

    // The character encoding the you want to use for the subject line and
    // message body of the email.
    public static String charset = "UTF-8";

    // The body of the email for recipients whose email clients support HTML
    content.
    static final String body = ""
        Amazon Pinpoint test (AWS SDK for Java 2.x)

        This email was sent through the Amazon Pinpoint Email API using the AWS SDK
    for Java 2.x

    """;

    public static void main(String[] args) {
        final String usage = ""

            Usage:    <subject> <appId> <senderAddress>
<toAddress>

            Where:
                subject - The email subject to use.
                senderAddress - The from address. This address has to be verified in
Amazon Pinpoint in the region you're using to send email\s
                toAddress - The to address. This address has to be verified in Amazon
Pinpoint in the region you're using to send email\s
            """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }
    }
}

```

```
String subject = args[0];
String senderAddress = args[1];
String toAddress = args[2];
System.out.println("Sending a message");
PinpointEmailClient pinpoint = PinpointEmailClient.builder()
    .region(Region.US_EAST_1)
    .build();

sendEmail(pinpoint, subject, senderAddress, toAddress);
System.out.println("Email was sent");
pinpoint.close();
}

public static void sendEmail(PinpointEmailClient pinpointEmailClient, String
subject, String senderAddress, String toAddress) {
    try {
        Content content = Content.builder()
            .data(body)
            .build();

        Body messageBody = Body.builder()
            .text(content)
            .build();

        Message message = Message.builder()
            .body(messageBody)
            .subject(Content.builder().data(subject).build())
            .build();

        Destination destination = Destination.builder()
            .toAddresses(toAddress)
            .build();

        EmailContent emailContent = EmailContent.builder()
            .simple(message)
            .build();

        SendEmailRequest sendEmailRequest = SendEmailRequest.builder()
            .fromEmailAddress(senderAddress)
            .destination(destination)
            .content(emailContent)
            .build();
```

```

        pinpointEmailClient.sendEmail(sendEmailRequest);
        System.out.println("Message Sent");

    } catch (PinpointException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}

```

### Invio di un messaggio e-mail con valori CC.

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;
import software.amazon.awssdk.services.pinpointemail.PinpointEmailClient;
import software.amazon.awssdk.services.pinpointemail.model.Body;
import software.amazon.awssdk.services.pinpointemail.model.Content;
import software.amazon.awssdk.services.pinpointemail.model.Destination;
import software.amazon.awssdk.services.pinpointemail.model.EmailContent;
import software.amazon.awssdk.services.pinpointemail.model.Message;
import software.amazon.awssdk.services.pinpointemail.model.SendEmailRequest;
import java.util.ArrayList;

/**
 * Before running this Java V2 code example, set up your development environment,
 * including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class SendEmailMessageCC {

    // The body of the email.
    static final String body = """"
        Amazon Pinpoint test (AWS SDK for Java 2.x)

        This email was sent through the Amazon Pinpoint Email API using the AWS SDK
        for Java 2.x

        """";

    public static void main(String[] args) {

```

```
final String usage = ""

Usage:    <subject> <senderAddress> <toAddress> <ccAddress>

Where:
    subject - The email subject to use.
    senderAddress - The from address. This address has to be verified in
Amazon Pinpoint in the region you're using to send email\s
    toAddress - The to address. This address has to be verified in Amazon
Pinpoint in the region you're using to send email\s
    ccAddress - The CC address.
"";

if (args.length != 4) {
    System.out.println(usage);
    System.exit(1);
}

String subject = args[0];
String senderAddress = args[1];
String toAddress = args[2];
String ccAddress = args[3];

System.out.println("Sending a message");
PinpointEmailClient pinpoint = PinpointEmailClient.builder()
    .region(Region.US_EAST_1)
    .build();

ArrayList<String> ccList = new ArrayList<>();
ccList.add(ccAddress);
sendEmail(pinpoint, subject, senderAddress, toAddress, ccList);
pinpoint.close();
}

public static void sendEmail(PinpointEmailClient pinpointEmailClient, String
subject, String senderAddress, String toAddress, ArrayList<String> ccAddresses) {
    try {
        Content content = Content.builder()
            .data(body)
            .build();

        Body messageBody = Body.builder()
            .text(content)
            .build();
```



```
Message message = Message.builder()
    .body(messageBody)
    .subject(Content.builder().data(subject).build())
    .build();

Destination destination = Destination.builder()
    .toAddresses(toAddress)
    .ccAddresses(ccAddresses)
    .build();

EmailContent emailContent = EmailContent.builder()
    .simple(message)
    .build();

SendEmailRequest sendEmailRequest = SendEmailRequest.builder()
    .fromEmailAddress(senderAddress)
    .destination(destination)
    .content(emailContent)
    .build();

pinpointEmailClient.sendEmail(sendEmailRequest);
System.out.println("Message Sent");

} catch (PinpointException e) {
    // Handle exception
    e.printStackTrace();
}
}
```

Invia un messaggio SMS.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.DirectMessageConfiguration;
import software.amazon.awssdk.services.pinpoint.model.SMSMessage;
import software.amazon.awssdk.services.pinpoint.model.AddressConfiguration;
import software.amazon.awssdk.services.pinpoint.model.ChannelType;
import software.amazon.awssdk.services.pinpoint.model.MessageRequest;
import software.amazon.awssdk.services.pinpoint.model.SendMessagesRequest;
import software.amazon.awssdk.services.pinpoint.model.SendMessagesResponse;
```

```
import software.amazon.awssdk.services.pinpoint.model.MessageResponse;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;
import java.util.HashMap;
import java.util.Map;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class SendMessage {

    // The type of SMS message that you want to send. If you plan to send
    // time-sensitive content, specify TRANSACTIONAL. If you plan to send
    // marketing-related content, specify PROMOTIONAL.
    public static String messageType = "TRANSACTIONAL";

    // The registered keyword associated with the originating short code.
    public static String registeredKeyword = "myKeyword";

    // The sender ID to use when sending the message. Support for sender ID
    // varies by country or region. For more information, see
    // https://docs.aws.amazon.com/pinpoint/latest/userguide/channels-sms-countries.html
    public static String senderId = "MySenderId";

    public static void main(String[] args) {
        final String usage = ""

            Usage:  <message> <appId> <originationNumber>
<destinationNumber>\s

            Where:
                message - The body of the message to send.
                appId - The Amazon Pinpoint project/application ID
to use when you send this message.
                originationNumber - The phone number or short code
that you specify has to be associated with your Amazon Pinpoint account. For best
results, specify long codes in E.164 format (for example, +1-555-555-5654).
```

destinationNumber - The recipient's phone number.  
 For best results, you should specify the phone number in E.164 format (for example, +1-555-555-5654).\s

```

        """;

    if (args.length != 4) {
        System.out.println(usage);
        System.exit(1);
    }

    String message = args[0];
    String appId = args[1];
    String originationNumber = args[2];
    String destinationNumber = args[3];
    System.out.println("Sending a message");
    PinpointClient pinpoint = PinpointClient.builder()
        .region(Region.US_EAST_1)
        .build();

    sendSMSMessage(pinpoint, message, appId, originationNumber,
destinationNumber);
    pinpoint.close();
}

public static void sendSMSMessage(PinpointClient pinpoint, String message,
String appId,
    String originationNumber,
    String destinationNumber) {
    try {
        Map<String, AddressConfiguration> addressMap = new
HashMap<String, AddressConfiguration>();
        AddressConfiguration addConfig =
AddressConfiguration.builder()

            .channelType(ChannelType.SMS)
            .build();

        addressMap.put(destinationNumber, addConfig);
        SMSMessage smsMessage = SMSMessage.builder()
            .body(message)
            .messageType(messageType)
            .originationNumber(originationNumber)
            .senderId(senderId)
            .keyword(registeredKeyword)
            .build();
    }
}

```

```

        // Create a DirectMessageConfiguration object.
        DirectMessageConfiguration direct =
DirectMessageConfiguration.builder()
            .smsMessage(smsMessage)
            .build();

        MessageRequest msgReq = MessageRequest.builder()
            .addresses(addressMap)
            .messageConfiguration(direct)
            .build();

        // create a SendMessagesRequest object
        SendMessagesRequest request = SendMessagesRequest.builder()
            .applicationId(appId)
            .messageRequest(msgReq)
            .build();

        SendMessagesResponse response =
pinpoint.sendMessage(request);
        MessageResponse msg1 = response.getMessageResponse();
        Map map1 = msg1.getResult();

        // Write out the result of sendMessage.
        map1.forEach((k, v) -> System.out.println((k + ":" + v)));

    } catch (PinpointException e) {
        System.err.println(e.getAwsErrorDetails().getErrorMessage());
        System.exit(1);
    }
}
}
}

```

Invia messaggi SMS in batch.

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.DirectMessageConfiguration;
import software.amazon.awssdk.services.pinpoint.model.SMSMessage;
import software.amazon.awssdk.services.pinpoint.model.AddressConfiguration;
import software.amazon.awssdk.services.pinpoint.model.ChannelType;
import software.amazon.awssdk.services.pinpoint.model.MessageRequest;

```

```
import software.amazon.awssdk.services.pinpoint.model.SendMessageRequest;
import software.amazon.awssdk.services.pinpoint.model.SendMessageResponse;
import software.amazon.awssdk.services.pinpoint.model.MessageResponse;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;
import java.util.HashMap;
import java.util.Map;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class SendMessageBatch {

    // The type of SMS message that you want to send. If you plan to send
    // time-sensitive content, specify TRANSACTIONAL. If you plan to send
    // marketing-related content, specify PROMOTIONAL.
    public static String messageType = "TRANSACTIONAL";

    // The registered keyword associated with the originating short code.
    public static String registeredKeyword = "myKeyword";

    // The sender ID to use when sending the message. Support for sender ID
    // varies by country or region. For more information, see
    // https://docs.aws.amazon.com/pinpoint/latest/userguide/channels-sms-countries.html
    public static String senderId = "MySenderId";

    public static void main(String[] args) {
        final String usage = ""

            Usage:  <message> <appId> <originationNumber>
<destinationNumber> <destinationNumber1>\s

            Where:
                message - The body of the message to send.
                appId - The Amazon Pinpoint project/application ID
to use when you send this message.
                originationNumber - The phone number or short code
that you specify has to be associated with your Amazon Pinpoint account. For best
results, specify long codes in E.164 format (for example, +1-555-555-5654).
```

destinationNumber - The recipient's phone number. For best results, you should specify the phone number in E.164 format (for example, +1-555-555-5654).

destinationNumber1 - The second recipient's phone number. For best results, you should specify the phone number in E.164 format (for example, +1-555-555-5654).\s

```

        """;

        if (args.length != 5) {
            System.out.println(usage);
            System.exit(1);
        }

        String message = args[0];
        String appId = args[1];
        String originationNumber = args[2];
        String destinationNumber = args[3];
        String destinationNumber1 = args[4];
        System.out.println("Sending a message");
        PinpointClient pinpoint = PinpointClient.builder()
            .region(Region.US_EAST_1)
            .build();

        sendSMSMessage(pinpoint, message, appId, originationNumber,
            destinationNumber, destinationNumber1);
        pinpoint.close();
    }

    public static void sendSMSMessage(PinpointClient pinpoint, String message,
        String appId,
            String originationNumber,
            String destinationNumber, String destinationNumber1) {
        try {
            Map<String, AddressConfiguration> addressMap = new
            HashMap<String, AddressConfiguration>();
            AddressConfiguration addConfig =
            AddressConfiguration.builder()
                .channelType(ChannelType.SMS)
                .build();

            // Add an entry to the Map object for each number to whom
            you want to send a
            // message.
            addressMap.put(destinationNumber, addConfig);

```

```
addressMap.put(destinationNumber1, addConfig);
SMSMessage smsMessage = SMSMessage.builder()
    .body(message)
    .messageType(messageType)
    .originationNumber(originationNumber)
    .senderId(senderId)
    .keyword(registeredKeyword)
    .build();

// Create a DirectMessageConfiguration object.
DirectMessageConfiguration direct =
DirectMessageConfiguration.builder()
    .smsMessage(smsMessage)
    .build();

MessageRequest msgReq = MessageRequest.builder()
    .addresses(addressMap)
    .messageConfiguration(direct)
    .build();

// Create a SendMessagesRequest object.
SendMessagesRequest request = SendMessagesRequest.builder()
    .applicationId(appId)
    .messageRequest(msgReq)
    .build();

SendMessageResponse response =
pinpoint.sendMessage(request);
MessageResponse msg1 = response.getMessageResponse();
Map map1 = msg1.getResult();

// Write out the result of sendMessage.
map1.forEach((k, v) -> System.out.println((k + ":" + v)));

} catch (PinpointException e) {
    System.err.println(e.getAwsErrorDetails().getErrorMessage());
    System.exit(1);
}
}
```

- Per i dettagli sull'API, consulta la [SendMessages](#) sezione AWS SDK for Java 2.x API Reference.

## UpdateEndpoint

Il seguente esempio di codice mostra come utilizzare `UpdateEndpoint`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.EndpointResponse;
import software.amazon.awssdk.services.pinpoint.model.EndpointRequest;
import software.amazon.awssdk.services.pinpoint.model.UpdateEndpointRequest;
import software.amazon.awssdk.services.pinpoint.model.UpdateEndpointResponse;
import software.amazon.awssdk.services.pinpoint.model.GetEndpointRequest;
import software.amazon.awssdk.services.pinpoint.model.GetEndpointResponse;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;
import software.amazon.awssdk.services.pinpoint.model.EndpointDemographic;
import software.amazon.awssdk.services.pinpoint.model.EndpointLocation;
import software.amazon.awssdk.services.pinpoint.model.EndpointUser;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.List;
import java.util.UUID;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Map;
import java.util.Date;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class UpdateEndpoint {
    public static void main(String[] args) {
```



```
final String usage = ""

    Usage: <appId>

    Where:
        appId - The ID of the application to create an endpoint for.

    """;

if (args.length != 1) {
    System.out.println(usage);
    System.exit(1);
}

String appId = args[0];
PinpointClient pinpoint = PinpointClient.builder()
    .region(Region.US_EAST_1)
    .build();

EndpointResponse response = createEndpoint(pinpoint, appId);
System.out.println("Got Endpoint: " + response.id());
pinpoint.close();
}

public static EndpointResponse createEndpoint(PinpointClient client, String
appId) {
    String endpointId = UUID.randomUUID().toString();
    System.out.println("Endpoint ID: " + endpointId);

    try {
        EndpointRequest endpointRequest = createEndpointRequestData();
        UpdateEndpointRequest updateEndpointRequest =
UpdateEndpointRequest.builder()
            .applicationId(appId)
            .endpointId(endpointId)
            .endpointRequest(endpointRequest)
            .build();

        UpdateEndpointResponse updateEndpointResponse =
client.updateEndpoint(updateEndpointRequest);
        System.out.println("Update Endpoint Response: " +
updateEndpointResponse.messageBody());

        GetEndpointRequest getEndpointRequest = GetEndpointRequest.builder()
```

```
        .applicationId(appId)
        .endpointId(endpointId)
        .build();

    GetEndpointResponse getEndpointResponse =
client.getEndpoint(getEndpointRequest);
    System.out.println(getEndpointResponse.endpointResponse().address());

System.out.println(getEndpointResponse.endpointResponse().channelType());

System.out.println(getEndpointResponse.endpointResponse().applicationId());

System.out.println(getEndpointResponse.endpointResponse().endpointStatus());
    System.out.println(getEndpointResponse.endpointResponse().requestId());
    System.out.println(getEndpointResponse.endpointResponse().user());

    return getEndpointResponse.endpointResponse();

} catch (PinpointException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
return null;
}

private static EndpointRequest createEndpointRequestData() {
    try {
        List<String> favoriteTeams = new ArrayList<>();
        favoriteTeams.add("Lakers");
        favoriteTeams.add("Warriors");
        HashMap<String, List<String>> customAttributes = new HashMap<>();
        customAttributes.put("team", favoriteTeams);

        EndpointDemographic demographic = EndpointDemographic.builder()
            .appVersion("1.0")
            .make("apple")
            .model("iPhone")
            .modelVersion("7")
            .platform("ios")
            .platformVersion("10.1.1")
            .timezone("America/Los_Angeles")
            .build();

        EndpointLocation location = EndpointLocation.builder()
```

```
        .city("Los Angeles")
        .country("US")
        .latitude(34.0)
        .longitude(-118.2)
        .postalCode("90068")
        .region("CA")
        .build();

    Map<String, Double> metrics = new HashMap<>();
    metrics.put("health", 100.00);
    metrics.put("luck", 75.00);

    EndpointUser user = EndpointUser.builder()
        .userId(UUID.randomUUID().toString())
        .build();

    DateFormat df = new SimpleDateFormat("yyyy-MM-dd'T'HH:mm'Z'"); // Quoted
    "Z" to indicate UTC, no timezone                                     // offset

    String nowAsISO = df.format(new Date());

    return EndpointRequest.builder()
        .address(UUID.randomUUID().toString())
        .attributes(customAttributes)
        .channelType("APNS")
        .demographic(demographic)
        .effectiveDate(nowAsISO)
        .location(location)
        .metrics(metrics)
        .optOut("NONE")
        .requestId(UUID.randomUUID().toString())
        .user(user)
        .build();

    } catch (PinpointException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return null;
}
}
```

- Per i dettagli sull'API, consulta la [UpdateEndpoint](#) sezione AWS SDK for Java 2.x API Reference.

## Esempi di API Amazon Pinpoint SMS e Voice con SDK for Java 2.x

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando l'API AWS SDK for Java 2.x con Amazon Pinpoint SMS and Voice.

Le operazioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le operazioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati e negli esempi tra servizi.

Scenari: esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio.

Ogni esempio include un collegamento a GitHub, dove puoi trovare istruzioni su come configurare ed eseguire il codice nel contesto.

### Argomenti

- [Azioni](#)

### Azioni

#### SendVoiceMessage

Il seguente esempio di codice mostra come utilizzare `SendVoiceMessage`.

#### SDK per Java 2.x

##### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.core.client.config.ClientOverrideConfiguration;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpointsmsvoice.PinpointSmsVoiceClient;
import software.amazon.awssdk.services.pinpointsmsvoice.model.SSMLMessageType;
import software.amazon.awssdk.services.pinpointsmsvoice.model.VoiceMessageContent;
```

```
import
    software.amazon.awssdk.services.pinpointsmsvoice.model.SendVoiceMessageRequest;
import
    software.amazon.awssdk.services.pinpointsmsvoice.model.PinpointSmsVoiceException;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class SendVoiceMessage {

    // The Amazon Polly voice that you want to use to send the message. For a
    list
    // of voices, see https://docs.aws.amazon.com/polly/latest/dg/voicelist.html
    static final String voiceName = "Matthew";

    // The language to use when sending the message. For a list of supported
    // languages, see
    // https://docs.aws.amazon.com/polly/latest/dg/SupportedLanguage.html
    static final String languageCode = "en-US";

    // The content of the message. This example uses SSML to customize and
    control
    // certain aspects of the message, such as by adding pauses and changing
    // phonation. The message can't contain any line breaks.
    static final String ssmlMessage = "<speak>This is a test message sent from "
        + "<emphasis>Amazon Pinpoint</emphasis> "
        + "using the <break strength='weak'/>AWS "
        + "SDK for Java. "
        + "<amazon:effect phonation='soft'>Thank "
        + "you for listening.</amazon:effect></speak>";

    public static void main(String[] args) {

        final String usage = ""
```

Usage: <originationNumber> <destinationNumber>\s

Where:

originationNumber - The phone number or short code that you specify has to be associated with your Amazon Pinpoint account. For best results, specify long codes in E.164 format (for example, +1-555-555-5654).

destinationNumber - The recipient's phone number. For best results, you should specify the phone number in E.164 format (for example, +1-555-555-5654).\s

""";

```

    if (args.length != 2) {
        System.out.println(usage);
        System.exit(1);
    }

    String originationNumber = args[0];
    String destinationNumber = args[1];
    System.out.println("Sending a voice message");

    // Set the content type to application/json.
    List<String> listVal = new ArrayList<>();
    listVal.add("application/json");
    Map<String, List<String>> values = new HashMap<>();
    values.put("Content-Type", listVal);

    ClientOverrideConfiguration config2 =
ClientOverrideConfiguration.builder()
        .headers(values)
        .build();

    PinpointSmsVoiceClient client = PinpointSmsVoiceClient.builder()
        .overrideConfiguration(config2)
        .region(Region.US_EAST_1)
        .build();

    sendVoiceMsg(client, originationNumber, destinationNumber);
    client.close();
}

public static void sendVoiceMsg(PinpointSmsVoiceClient client, String
originationNumber,
        String destinationNumber) {
    try {

```

```

        SSMLMessageType ssmlMessageType = SSMLMessageType.builder()
            .languageCode(languageCode)
            .text(ssmlMessage)
            .voiceId(voiceName)
            .build();

        VoiceMessageContent content = VoiceMessageContent.builder()
            .ssmlMessage(ssmlMessageType)
            .build();

        SendVoiceMessageRequest voiceMessageRequest =
SendVoiceMessageRequest.builder()
            .destinationPhoneNumber(destinationNumber)
            .originationPhoneNumber(originationNumber)
            .content(content)
            .build();

        client.sendVoiceMessage(voiceMessageRequest);
        System.out.println("The message was sent successfully.");

    } catch (PinpointSmsVoiceException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
}

```

- Per i dettagli sull'API, consulta la [SendVoiceMessage](#) sezione AWS SDK for Java 2.x API Reference.

## Esempi di Amazon Polly con SDK for Java 2.x

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando Amazon Polly. AWS SDK for Java 2.x

Le operazioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le operazioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati e negli esempi tra servizi.

Scenari: esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio.

Ogni esempio include un collegamento a GitHub, dove puoi trovare istruzioni su come configurare ed eseguire il codice nel contesto.

## Argomenti

- [Azioni](#)

## Azioni

### DescribeVoices

Il seguente esempio di codice mostra come utilizzare `DescribeVoices`.

#### SDK per Java 2.x

#### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.polly.PollyClient;
import software.amazon.awssdk.services.polly.model.DescribeVoicesRequest;
import software.amazon.awssdk.services.polly.model.DescribeVoicesResponse;
import software.amazon.awssdk.services.polly.model.PollyException;
import software.amazon.awssdk.services.polly.model.Voice;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DescribeVoicesSample {
    public static void main(String args[]) {
        PollyClient polly = PollyClient.builder()
            .region(Region.US_WEST_2)
            .build();
    }
}
```



```
        describeVoice(polly);
        polly.close();
    }

    public static void describeVoice(PollyClient polly) {
        try {
            DescribeVoicesRequest voicesRequest = DescribeVoicesRequest.builder()
                .languageCode("en-US")
                .build();

            DescribeVoicesResponse enUsVoicesResult =
polly.describeVoices(voicesRequest);
            List<Voice> voices = enUsVoicesResult.voices();
            for (Voice myVoice : voices) {
                System.out.println("The ID of the voice is " + myVoice.id());
                System.out.println("The gender of the voice is " +
myVoice.gender());
            }

        } catch (PollyException e) {
            System.err.println("Exception caught: " + e);
            System.exit(1);
        }
    }
}
```

- Per i dettagli sull'API, consulta la [DescribeVoices](#) sezione AWS SDK for Java 2.x API Reference.

## ListLexicons

Il seguente esempio di codice mostra come utilizzare `ListLexicons`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.polly.PollyClient;
import software.amazon.awssdk.services.polly.model.ListLexiconsResponse;
import software.amazon.awssdk.services.polly.model.ListLexiconsRequest;
import software.amazon.awssdk.services.polly.model.LexiconDescription;
import software.amazon.awssdk.services.polly.model.PollyException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListLexicons {
    public static void main(String args[]) {
        PollyClient polly = PollyClient.builder()
            .region(Region.US_WEST_2)
            .build();

        listLexicons(polly);
        polly.close();
    }

    public static void listLexicons(PollyClient client) {
        try {
            ListLexiconsRequest listLexiconsRequest = ListLexiconsRequest.builder()
                .build();

            ListLexiconsResponse listLexiconsResult =
client.listLexicons(listLexiconsRequest);
            List<LexiconDescription> lexiconDescription =
listLexiconsResult.lexicons();
            for (LexiconDescription lexDescription : lexiconDescription) {
                System.out.println("The name of the Lexicon is " +
lexDescription.name());
            }

        } catch (PollyException e) {
            System.err.println("Exception caught: " + e);
            System.exit(1);
        }
    }
}
```

```

    }
  }
}

```

- Per i dettagli sull'API, consulta la [ListLexicons](#) sezione AWS SDK for Java 2.x API Reference.

## SynthesizeSpeech

Il seguente esempio di codice mostra come utilizzare `SynthesizeSpeech`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```

import javazoom.jl.decoder.JavaLayerException;
import software.amazon.awssdk.core.ResponseInputStream;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.polly.PollyClient;
import software.amazon.awssdk.services.polly.model.DescribeVoicesRequest;
import software.amazon.awssdk.services.polly.model.Voice;
import software.amazon.awssdk.services.polly.model.DescribeVoicesResponse;
import software.amazon.awssdk.services.polly.model.OutputFormat;
import software.amazon.awssdk.services.polly.model.PollyException;
import software.amazon.awssdk.services.polly.model.SynthesizeSpeechRequest;
import software.amazon.awssdk.services.polly.model.SynthesizeSpeechResponse;
import java.io.IOException;
import java.io.InputStream;
import javazoom.jl.player.advanced.AdvancedPlayer;
import javazoom.jl.player.advanced.PlaybackEvent;
import javazoom.jl.player.advanced.PlaybackListener;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 */

```

```
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class PollyDemo {
    private static final String SAMPLE = "Congratulations. You have successfully
built this working demo " +
        " of Amazon Polly in Java Version 2. Have fun building voice enabled
apps with Amazon Polly (that's me!), and always "
    +
        " look at the AWS website for tips and tricks on using Amazon Polly and
other great services from AWS";

    public static void main(String args[]) {
        PollyClient polly = PollyClient.builder()
            .region(Region.US_WEST_2)
            .build();

        talkPolly(polly);
        polly.close();
    }

    public static void talkPolly(PollyClient polly) {
        try {
            DescribeVoicesRequest describeVoiceRequest =
DescribeVoicesRequest.builder()
                .engine("standard")
                .build();

            DescribeVoicesResponse describeVoicesResult =
polly.describeVoices(describeVoiceRequest);
            Voice voice = describeVoicesResult.voices().stream()
                .filter(v -> v.name().equals("Joanna"))
                .findFirst()
                .orElseThrow(() -> new RuntimeException("Voice not found"));
            InputStream stream = synthesize(polly, SAMPLE, voice, OutputFormat.MP3);
            AdvancedPlayer player = new AdvancedPlayer(stream,

javazoom.jl.player.FactoryRegistry.systemRegistry().createAudioDevice());
            player.setPlayBackListener(new PlaybackListener() {
                public void playbackStarted(PlaybackEvent evt) {
                    System.out.println("Playback started");
                    System.out.println(SAMPLE);
                }
            });

            public void playbackFinished(PlaybackEvent evt) {
```

```
        System.out.println("Playback finished");
    }
});

// play it!
player.play();

} catch (PollyException | JavaLayerException | IOException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
}

public static InputStream synthesize(PollyClient polly, String text, Voice
voice, OutputFormat format)
    throws IOException {
    SynthesizeSpeechRequest synthReq = SynthesizeSpeechRequest.builder()
        .text(text)
        .voiceId(voice.id())
        .outputFormat(format)
        .build();

    ResponseInputStream<SynthesizeSpeechResponse> synthRes =
polly.synthesizeSpeech(synthReq);
    return synthRes;
}
}
```

- Per i dettagli sull'API, consulta la [SynthesizeSpeech](#) sezione AWS SDK for Java 2.x API Reference.

## Esempi di Amazon RDS che utilizzano SDK for Java 2.x

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando Amazon RDS. AWS SDK for Java 2.x

Le operazioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le operazioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati e negli esempi tra servizi.

Scenari: esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio.


Ogni esempio include un collegamento a GitHub, dove puoi trovare istruzioni su come configurare ed eseguire il codice nel contesto.

Nozioni di base

Hello Amazon RDS

Gli esempi di codice seguenti mostrano come iniziare a utilizzare Amazon RDS.

SDK per Java 2.x

 Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rds.RdsClient;
import software.amazon.awssdk.services.rds.model.DescribeDbInstancesResponse;
import software.amazon.awssdk.services.rds.model.DBInstance;
import software.amazon.awssdk.services.rds.model.RdsException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DescribeDBInstances {

    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        RdsClient rdsClient = RdsClient.builder()
            .region(region)
            .build();
```

```
        describeInstances(rdsClient);
        rdsClient.close();
    }

    public static void describeInstances(RdsClient rdsClient) {
        try {
            DescribeDbInstancesResponse response = rdsClient.describeDBInstances();
            List<DBInstance> instanceList = response.dbInstances();
            for (DBInstance instance : instanceList) {
                System.out.println("Instance ARN is: " + instance.dbInstanceArn());
                System.out.println("The Engine is " + instance.engine());
                System.out.println("Connection endpoint is" +
instance.endpoint().address());
            }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
}
```

- Per informazioni dettagliate sull'API, consulta [DescribeDBInstances](#) nella Documentazione di riferimento delle API di AWS SDK for Java 2.x .

## Argomenti

- [Azioni](#)
- [Scenari](#)

## Azioni

### **CreateDBInstance**

Il seguente esempio di codice mostra come usare `CreateDBInstance`.

## SDK per Java 2.x

 Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import com.google.gson.Gson;
import
    software.amazon.awssdk.auth.credentials.EnvironmentVariableCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rds.RdsClient;
import software.amazon.awssdk.services.rds.model.DescribeDbInstancesRequest;
import software.amazon.awssdk.services.rds.model.CreateDbInstanceRequest;
import software.amazon.awssdk.services.rds.model.CreateDbInstanceResponse;
import software.amazon.awssdk.services.rds.model.RdsException;
import software.amazon.awssdk.services.rds.model.DescribeDbInstancesResponse;
import software.amazon.awssdk.services.rds.model.DBInstance;
import software.amazon.awssdk.services.secretsmanager.SecretsManagerClient;
import software.amazon.awssdk.services.secretsmanager.model.GetSecretValueRequest;
import software.amazon.awssdk.services.secretsmanager.model.GetSecretValueResponse;

import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * This example requires an AWS Secrets Manager secret that contains the
 * database credentials. If you do not create a
 * secret, this example will not work. For more details, see:
 *
 * https://docs.aws.amazon.com/secretsmanager/latest/userguide/integrating_how-
 * services-use-secrets_RS.html
 *
 */
```



```
public class CreateDBInstance {
    public static long sleepTime = 20;

    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <dbInstanceIdentifier> <dbName> <secretName>

            Where:
                dbInstanceIdentifier - The database instance identifier.\s
                dbName - The database name.\s
                secretName - The name of the AWS Secrets Manager secret that
contains the database credentials."
            """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String dbInstanceIdentifier = args[0];
        String dbName = args[1];
        String secretName = args[2];
        Gson gson = new Gson();
        User user = gson.fromJson(String.valueOf(getSecretValues(secretName)),
User.class);
        Region region = Region.US_WEST_2;
        RdsClient rdsClient = RdsClient.builder()
            .region(region)
            .build();

        createDatabaseInstance(rdsClient, dbInstanceIdentifier, dbName,
user.getUsername(), user.getPassword());
        waitForInstanceReady(rdsClient, dbInstanceIdentifier);
        rdsClient.close();
    }

    private static SecretsManagerClient getSecretClient() {
        Region region = Region.US_WEST_2;
        return SecretsManagerClient.builder()
            .region(region)
    }
}
```

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

private static String getSecretValues(String secretName) {
    SecretsManagerClient secretClient = getSecretClient();
    GetSecretValueRequest valueRequest = GetSecretValueRequest.builder()
        .secretId(secretName)
        .build();

    GetSecretValueResponse valueResponse =
secretClient.getSecretValue(valueRequest);
    return valueResponse.secretString();
}

public static void createDatabaseInstance(RdsClient rdsClient,
    String dbInstanceIdentifier,
    String dbName,
    String userName,
    String userPassword) {

    try {
        CreateDbInstanceRequest instanceRequest =
CreateDbInstanceRequest.builder()
            .dbInstanceIdentifier(dbInstanceIdentifier)
            .allocatedStorage(100)
            .dbName(dbName)
            .engine("mysql")
            .dbInstanceClass("db.m4.large")
            .engineVersion("8.0")
            .storageType("standard")
            .masterUsername(userName)
            .masterUserPassword(userPassword)
            .build();

        CreateDbInstanceResponse response =
rdsClient.createDBInstance(instanceRequest);
        System.out.print("The status is " +
response.dbInstance().dbInstanceStatus());

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}
```

```
    }  
  }  
  
  // Waits until the database instance is available.  
  public static void waitForInstanceReady(RdsClient rdsClient, String  
dbInstanceIdentifier) {  
    boolean instanceReady = false;  
    String instanceReadyStr;  
    System.out.println("Waiting for instance to become available.");  
    try {  
      DescribeDbInstancesRequest instanceRequest =  
DescribeDbInstancesRequest.builder()  
        .dbInstanceIdentifier(dbInstanceIdentifier)  
        .build();  
  
      // Loop until the cluster is ready.  
      while (!instanceReady) {  
        DescribeDbInstancesResponse response =  
rdsClient.describeDBInstances(instanceRequest);  
        List<DBInstance> instanceList = response.dbInstances();  
        for (DBInstance instance : instanceList) {  
          instanceReadyStr = instance.dbInstanceStatus();  
          if (instanceReadyStr.contains("available"))  
            instanceReady = true;  
          else {  
            System.out.print(".");  
            Thread.sleep(sleepTime * 1000);  
          }  
        }  
      }  
      System.out.println("Database instance is available!");  
  
    } catch (RdsException | InterruptedException e) {  
      System.err.println(e.getMessage());  
      System.exit(1);  
    }  
  }  
}
```

- Per informazioni dettagliate sull'API, consulta [CreateDBInstance](#) nella Documentazione di riferimento delle API di AWS SDK for Java 2.x .

## CreateDBParameterGroup

Il seguente esempio di codice mostra come usare `CreateDBParameterGroup`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public static void createDBParameterGroup(RdsClient rdsClient, String
dbGroupName, String dbParameterGroupFamily) {
    try {
        CreateDbParameterGroupRequest groupRequest =
CreateDbParameterGroupRequest.builder()
            .dbParameterGroupName(dbGroupName)
            .dbParameterGroupFamily(dbParameterGroupFamily)
            .description("Created by using the AWS SDK for Java")
            .build();

        CreateDbParameterGroupResponse response =
rdsClient.createDBParameterGroup(groupRequest);
        System.out.println("The group name is " +
response.dbParameterGroup().dbParameterGroupName());

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}
```

- Per i dettagli sull'API, consulta [CreateDB ParameterGroup](#) in AWS SDK for Java 2.x API Reference.

## CreateDBSnapshot

Il seguente esempio di codice mostra come utilizzare `CreateDBSnapshot`.

## SDK per Java 2.x

 Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
// Create an Amazon RDS snapshot.
public static void createSnapshot(RdsClient rdsClient, String
dbInstanceIdentifier, String dbSnapshotIdentifier) {
    try {
        CreateDbSnapshotRequest snapshotRequest =
CreateDbSnapshotRequest.builder()
            .dbInstanceIdentifier(dbInstanceIdentifier)
            .dbSnapshotIdentifier(dbSnapshotIdentifier)
            .build();

        CreateDbSnapshotResponse response =
rdsClient.createDBSnapshot(snapshotRequest);
        System.out.println("The Snapshot id is " +
response.dbSnapshot().dbiResourceId());

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}
```

- Per informazioni dettagliate sull'API, consulta [CreateDBSnapshot](#) nella Documentazione di riferimento dell'API AWS SDK for Java 2.x .

## DeleteDBInstance

Il seguente esempio di codice mostra come usare `DeleteDBInstance`.

## SDK per Java 2.x

 Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rds.RdsClient;
import software.amazon.awssdk.services.rds.model.DeleteDbInstanceRequest;
import software.amazon.awssdk.services.rds.model.DeleteDbInstanceResponse;
import software.amazon.awssdk.services.rds.model.RdsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteDBInstance {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <dbInstanceIdentifier>\s

            Where:
                dbInstanceIdentifier - The database instance identifier\s
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String dbInstanceIdentifier = args[0];
        Region region = Region.US_WEST_2;
        RdsClient rdsClient = RdsClient.builder()
            .region(region)
```

```
        .build();

        deleteDatabaseInstance(rdsClient, dbInstanceIdentifier);
        rdsClient.close();
    }

    public static void deleteDatabaseInstance(RdsClient rdsClient, String
dbInstanceIdentifier) {
        try {
            DeleteDbInstanceRequest deleteDbInstanceRequest =
DeleteDbInstanceRequest.builder()
                .dbInstanceIdentifier(dbInstanceIdentifier)
                .deleteAutomatedBackups(true)
                .skipFinalSnapshot(true)
                .build();

            DeleteDbInstanceResponse response =
rdsClient.deleteDBInstance(deleteDbInstanceRequest);
            System.out.print("The status of the database is " +
response.dbInstance().dbInstanceStatus());

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
}
```

- Per informazioni dettagliate sull'API, consulta [DeleteDBInstance](#) nella Documentazione di riferimento delle API di AWS SDK for Java 2.x .

## DeleteDBParameterGroup

Il seguente esempio di codice mostra come usare `DeleteDBParameterGroup`.

## SDK per Java 2.x

 Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
// Delete the parameter group after database has been deleted.
// An exception is thrown if you attempt to delete the para group while database
// exists.
public static void deleteParaGroup(RdsClient rdsClient, String dbGroupName,
String dbARN)
    throws InterruptedException {
    try {
        boolean isDataDel = false;
        boolean didFind;
        String instanceARN;

        // Make sure that the database has been deleted.
        while (!isDataDel) {
            DescribeDbInstancesResponse response =
rdsClient.describeDBInstances();
            List<DBInstance> instanceList = response.dbInstances();
            int listSize = instanceList.size();
            didFind = false;
            int index = 1;
            for (DBInstance instance : instanceList) {
                instanceARN = instance.dbInstanceArn();
                if (instanceARN.compareTo(dbARN) == 0) {
                    System.out.println(dbARN + " still exists");
                    didFind = true;
                }
            }
            if ((index == listSize) && (!didFind)) {
                // Went through the entire list and did not find the
database ARN.

                isDataDel = true;
            }
            Thread.sleep(sleepTime * 1000);
            index++;
        }
    }
}
```



```

        // Delete the para group.
        DeleteDbParameterGroupRequest parameterGroupRequest =
DeleteDbParameterGroupRequest.builder()
        .dbParameterGroupName(dbGroupName)
        .build();

        rdsClient.deleteDBParameterGroup(parameterGroupRequest);
        System.out.println(dbGroupName + " was deleted.");

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}

```

- Per i dettagli sull'API, consulta [DeleteDB ParameterGroup](#) in AWS SDK for Java 2.x API Reference.

## DescribeAccountAttributes

Il seguente esempio di codice mostra come utilizzare `DescribeAccountAttributes`

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rds.RdsClient;
import software.amazon.awssdk.services.rds.model.AccountQuota;
import software.amazon.awssdk.services.rds.model.RdsException;
import software.amazon.awssdk.services.rds.model.DescribeAccountAttributesResponse;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development

```

```
* environment, including your credentials.
*
* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class DescribeAccountAttributes {
    public static void main(String[] args) {
        Region region = Region.US_WEST_2;
        RdsClient rdsClient = RdsClient.builder()
            .region(region)
            .build();

        getAccountAttributes(rdsClient);
        rdsClient.close();
    }

    public static void getAccountAttributes(RdsClient rdsClient) {
        try {
            DescribeAccountAttributesResponse response =
rdsClient.describeAccountAttributes();
            List<AccountQuota> quotasList = response.accountQuotas();
            for (AccountQuota quotas : quotasList) {
                System.out.println("Name is: " + quotas.accountQuotaName());
                System.out.println("Max value is " + quotas.max());
            }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
}
```

- Per i dettagli sull'API, consulta la [DescribeAccountAttributes](#) sezione AWS SDK for Java 2.x API Reference.

## DescribeDBEngineVersions

Il seguente esempio di codice mostra come utilizzare `DescribeDBEngineVersions`.

## SDK per Java 2.x

 Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public static void describeDBEngines(RdsClient rdsClient) {
    try {
        DescribeDbEngineVersionsRequest engineVersionsRequest =
DescribeDbEngineVersionsRequest.builder()
            .defaultOnly(true)
            .engine("mysql")
            .maxRecords(20)
            .build();

        DescribeDbEngineVersionsResponse response =
rdsClient.describeDBEngineVersions(engineVersionsRequest);
        List<DBEngineVersion> engines = response.dbEngineVersions();

        // Get all DBEngineVersion objects.
        for (DBEngineVersion engineOb : engines) {
            System.out.println("The name of the DB parameter group family for
the database engine is "
                + engineOb.dbParameterGroupFamily());
            System.out.println("The name of the database engine " +
engineOb.engine());
            System.out.println("The version number of the database engine " +
engineOb.engineVersion());
        }

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}
```

- Per i dettagli sull'API, consulta [DescribeDB EngineVersions](#) in AWS SDK for Java 2.x API Reference.

## DescribeDBInstances

Il seguente esempio di codice mostra come utilizzare `DescribeDBInstances`

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rds.RdsClient;
import software.amazon.awssdk.services.rds.model.DescribeDbInstancesResponse;
import software.amazon.awssdk.services.rds.model.DBInstance;
import software.amazon.awssdk.services.rds.model.RdsException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DescribeDBInstances {

    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        RdsClient rdsClient = RdsClient.builder()
            .region(region)
            .build();

        describeInstances(rdsClient);
        rdsClient.close();
    }

    public static void describeInstances(RdsClient rdsClient) {
        try {
            DescribeDbInstancesResponse response = rdsClient.describeDBInstances();
            List<DBInstance> instanceList = response.dbInstances();
        }
    }
}
```

```
        for (DBInstance instance : instanceList) {
            System.out.println("Instance ARN is: " + instance.dbInstanceArn());
            System.out.println("The Engine is " + instance.engine());
            System.out.println("Connection endpoint is" +
instance.endpoint().address());
        }

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}
}
```

- Per informazioni dettagliate sull'API, consulta [DescribeDBInstances](#) nella Documentazione di riferimento delle API di AWS SDK for Java 2.x .

## DescribeDBParameterGroups

Il seguente esempio di codice mostra come usare `DescribeDBParameterGroups`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public static void describeDbParameterGroups(RdsClient rdsClient, String
dbGroupName) {
    try {
        DescribeDbParameterGroupsRequest groupsRequest =
DescribeDbParameterGroupsRequest.builder()
            .dbParameterGroupName(dbGroupName)
            .maxRecords(20)
            .build();

        DescribeDbParameterGroupsResponse response =
rdsClient.describeDBParameterGroups(groupsRequest);
```

```
List<DBParameterGroup> groups = response.dbParameterGroups();
for (DBParameterGroup group : groups) {
    System.out.println("The group name is " +
group.dbParameterGroupName());
    System.out.println("The group description is " +
group.description());
}

} catch (RdsException e) {
    System.out.println(e.getLocalizedMessage());
    System.exit(1);
}
}
```

- Per i dettagli sull'API, consulta [DescribeDB ParameterGroups](#) in AWS SDK for Java 2.x API Reference.

## DescribeDBParameters

Il seguente esempio di codice mostra come utilizzare `DescribeDBParameters`

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
// Retrieve parameters in the group.
public static void describeDbParameters(RdsClient rdsClient, String dbGroupName,
int flag) {
    try {
        DescribeDbParametersRequest dbParameterGroupsRequest;
        if (flag == 0) {
            dbParameterGroupsRequest = DescribeDbParametersRequest.builder()
                .dbParameterGroupName(dbGroupName)
                .build();
        } else {
            dbParameterGroupsRequest = DescribeDbParametersRequest.builder()
```

```

        .dbParameterGroupName(dbGroupName)
        .source("user")
        .build();
    }

    DescribeDbParametersResponse response =
rdsClient.describeDBParameters(dbParameterGroupsRequest);
    List<Parameter> dbParameters = response.parameters();
    String paraName;
    for (Parameter para : dbParameters) {
        // Only print out information about either auto_increment_offset or
        // auto_increment_increment.
        paraName = para.parameterName();
        if ((paraName.compareTo("auto_increment_offset") == 0)
            || (paraName.compareTo("auto_increment_increment ") == 0)) {
            System.out.println("*** The parameter name is " + paraName);
            System.out.println("*** The parameter value is " +
para.parameterValue());
            System.out.println("*** The parameter data type is " +
para.dataType());
            System.out.println("*** The parameter description is " +
para.description());
            System.out.println("*** The parameter allowed values is " +
para.allowedValues());
        }
    }

} catch (RdsException e) {
    System.out.println(e.getLocalizedMessage());
    System.exit(1);
}
}


```

- Per informazioni sull'API, consulta [DescribeDBParameters](#) nella Documentazione di riferimento dell'API AWS SDK for Java 2.x .

## DescribeOrderableDBInstanceOptions

Il seguente esempio di codice mostra come usare `DescribeOrderableDBInstanceOptions`.

## SDK per Java 2.x

 Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
// Get a list of allowed engine versions.
public static void getAllowedEngines(RdsClient rdsClient, String
dbParameterGroupFamily) {
    try {
        DescribeDbEngineVersionsRequest versionsRequest =
DescribeDbEngineVersionsRequest.builder()
            .dbParameterGroupFamily(dbParameterGroupFamily)
            .engine("mysql")
            .build();

        DescribeDbEngineVersionsResponse response =
rdsClient.describeDBEngineVersions(versionsRequest);
        List<DBEngineVersion> dbEngines = response.dbEngineVersions();
        for (DBEngineVersion dbEngine : dbEngines) {
            System.out.println("The engine version is " +
dbEngine.engineVersion());
            System.out.println("The engine description is " +
dbEngine.dbEngineDescription());
        }

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}
```

- Per i dettagli sull'API, consulta [DescribeOrderableDB InstanceOptions](#) in AWS SDK for Java 2.x API Reference.

## GenerateRDSAuthToken

Il seguente esempio di codice mostra come utilizzare `GenerateRDSAuthToken`.



## SDK per Java 2.x

 Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Usa la [RdsUtilities](#) classe per generare un token di autenticazione.

```
public class GenerateRDSAuthToken {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <dbInstanceIdentifier> <masterUsername>

            Where:
                dbInstanceIdentifier - The database instance identifier.\s
                masterUsername - The master user name.\s
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String dbInstanceIdentifier = args[0];
        String masterUsername = args[1];
        Region region = Region.US_WEST_2;
        RdsClient rdsClient = RdsClient.builder()
            .region(region)
            .build();

        String token = getAuthToken(rdsClient, dbInstanceIdentifier,
masterUsername);
        System.out.println("The token response is " + token);
    }

    public static String getAuthToken(RdsClient rdsClient, String
dbInstanceIdentifier, String masterUsername) {

        RdsUtilities utilities = rdsClient.utilities();
```

```
    try {
        GenerateAuthenticationTokenRequest tokenRequest =
GenerateAuthenticationTokenRequest.builder()
            .credentialsProvider(ProfileCredentialsProvider.create())
            .username(masterUsername)
            .port(3306)
            .hostname(dbInstanceIdentifier)
            .build();

        return utilities.generateAuthenticationToken(tokenRequest);

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
    return "";
}
}
```

- Per i dettagli sull'API, consulta [GenerateRDS AuthToken](#) in AWS SDK for Java 2.x API Reference.

## ModifyDBInstance

Il seguente esempio di codice mostra come utilizzare `ModifyDBInstance`

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rds.RdsClient;
import software.amazon.awssdk.services.rds.model.ModifyDbInstanceRequest;
import software.amazon.awssdk.services.rds.model.ModifyDbInstanceResponse;
import software.amazon.awssdk.services.rds.model.RdsException;
```

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ModifyDBInstance {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <dbInstanceIdentifier> <dbSnapshotIdentifier>\s
                Where:
                dbInstanceIdentifier - The database instance identifier.\s
                masterUserPassword - The updated password that corresponds to
the master user name.\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String dbInstanceIdentifier = args[0];
        String masterUserPassword = args[1];
        Region region = Region.US_WEST_2;
        RdsClient rdsClient = RdsClient.builder()
            .region(region)
            .build();

        updateIntance(rdsClient, dbInstanceIdentifier, masterUserPassword);
        rdsClient.close();
    }

    public static void updateIntance(RdsClient rdsClient, String
dbInstanceIdentifier, String masterUserPassword) {
        try {
            // For a demo - modify the DB instance by modifying the master password.
            ModifyDbInstanceRequest modifyDbInstanceRequest =
ModifyDbInstanceRequest.builder()
                .dbInstanceIdentifier(dbInstanceIdentifier)
                .publiclyAccessible(true)

```

```

        .masterUserPassword(masterUserPassword)
        .build();

        ModifyDbInstanceResponse instanceResponse =
rdsClient.modifyDBInstance(modifyDbInstanceRequest);
        System.out.print("The ARN of the modified database is: " +
instanceResponse.dbInstance().dbInstanceArn());

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}
}
}

```

- Per informazioni dettagliate sulle API, consulta [ModifyDBInstance](#) nella documentazione di riferimento dell'API AWS SDK for Java 2.x .

## ModifyDBParameterGroup

Il seguente esempio di codice mostra come usare `ModifyDBParameterGroup`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```

// Modify auto_increment_offset and auto_increment_increment parameters.
public static void modifyDBParas(RdsClient rdsClient, String dbGroupName) {
    try {
        Parameter parameter1 = Parameter.builder()
            .parameterName("auto_increment_offset")
            .applyMethod("immediate")
            .parameterValue("5")
            .build();

        List<Parameter> paraList = new ArrayList<>();
    }
}

```

```
        paraList.add(parameter1);
        ModifyDbParameterGroupRequest groupRequest =
ModifyDbParameterGroupRequest.builder()
            .dbParameterGroupName(dbGroupName)
            .parameters(paraList)
            .build();

        ModifyDbParameterGroupResponse response =
rdsClient.modifyDBParameterGroup(groupRequest);
        System.out.println("The parameter group " +
response.dbParameterGroupName() + " was successfully modified");

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}
```

- Per i dettagli sull'API, consulta [ModifyDB ParameterGroup](#) in AWS SDK for Java 2.x API Reference.

## RebootDBInstance

Il seguente esempio di codice mostra come utilizzare `RebootDBInstance`

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rds.RdsClient;
import software.amazon.awssdk.services.rds.model.RebootDbInstanceRequest;
import software.amazon.awssdk.services.rds.model.RebootDbInstanceResponse;
import software.amazon.awssdk.services.rds.model.RdsException;

/**
```

```
* Before running this Java V2 code example, set up your development
* environment, including your credentials.
*
* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class RebootDBInstance {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <dbInstanceIdentifier>\s

            Where:
                dbInstanceIdentifier - The database instance identifier\s
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String dbInstanceIdentifier = args[0];
        Region region = Region.US_WEST_2;
        RdsClient rdsClient = RdsClient.builder()
            .region(region)
            .build();

        rebootInstance(rdsClient, dbInstanceIdentifier);
        rdsClient.close();
    }

    public static void rebootInstance(RdsClient rdsClient, String
dbInstanceIdentifier) {
        try {
            RebootDbInstanceRequest rebootDbInstanceRequest =
RebootDbInstanceRequest.builder()
                .dbInstanceIdentifier(dbInstanceIdentifier)
                .build();

            RebootDbInstanceResponse instanceResponse =
rdsClient.rebootDBInstance(rebootDbInstanceRequest);
```

```
        System.out.print("The database " +
instanceResponse.dbInstance().dbInstanceArn() + " was rebooted");

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
}
```

- Per informazioni dettagliate sulle API, consulta [RebootDBInstance](#) nella documentazione di riferimento dell'API AWS SDK for Java 2.x .

## Scenari

### Nozioni di base sulle istanze DB

L'esempio di codice seguente mostra come:

- Creare un gruppo di parametri database personalizzati e imposta i relativi valori.
- Creare un'istanza database configurata per utilizzare il gruppo di parametri. L'istanza DB contiene anche un database.
- Acquisire uno snapshot dell'istanza.
- Eliminare l'istanza e il gruppo di parametri.

### SDK per Java 2.x

#### Note

C'è dell'altro GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Esegui più operazioni.

```
import com.google.gson.Gson;
import
software.amazon.awssdk.auth.credentials.EnvironmentVariableCredentialsProvider;
```

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rds.RdsClient;
import software.amazon.awssdk.services.rds.model.CreateDbInstanceRequest;
import software.amazon.awssdk.services.rds.model.CreateDbInstanceResponse;
import software.amazon.awssdk.services.rds.model.CreateDbParameterGroupResponse;
import software.amazon.awssdk.services.rds.model.CreateDbSnapshotRequest;
import software.amazon.awssdk.services.rds.model.CreateDbSnapshotResponse;
import software.amazon.awssdk.services.rds.model.DBEngineVersion;
import software.amazon.awssdk.services.rds.model.DBInstance;
import software.amazon.awssdk.services.rds.model.DBParameterGroup;
import software.amazon.awssdk.services.rds.model.DBSnapshot;
import software.amazon.awssdk.services.rds.model.DeleteDbInstanceRequest;
import software.amazon.awssdk.services.rds.model.DeleteDbInstanceResponse;
import software.amazon.awssdk.services.rds.model.DescribeDbEngineVersionsRequest;
import software.amazon.awssdk.services.rds.model.DescribeDbEngineVersionsResponse;
import software.amazon.awssdk.services.rds.model.DescribeDbInstancesRequest;
import software.amazon.awssdk.services.rds.model.DescribeDbInstancesResponse;
import software.amazon.awssdk.services.rds.model.DescribeDbParameterGroupsResponse;
import software.amazon.awssdk.services.rds.model.DescribeDbParametersResponse;
import software.amazon.awssdk.services.rds.model.DescribeDbSnapshotsRequest;
import software.amazon.awssdk.services.rds.model.DescribeDbSnapshotsResponse;
import
    software.amazon.awssdk.services.rds.model.DescribeOrderableDbInstanceOptionsResponse;
import software.amazon.awssdk.services.rds.model.ModifyDbParameterGroupResponse;
import software.amazon.awssdk.services.rds.model.OrderableDBInstanceOption;
import software.amazon.awssdk.services.rds.model.Parameter;
import software.amazon.awssdk.services.rds.model.RdsException;
import software.amazon.awssdk.services.rds.model.CreateDbParameterGroupRequest;
import software.amazon.awssdk.services.rds.model.DescribeDbParameterGroupsRequest;
import software.amazon.awssdk.services.rds.model.DescribeDbParametersRequest;
import software.amazon.awssdk.services.rds.model.ModifyDbParameterGroupRequest;
import
    software.amazon.awssdk.services.rds.model.DescribeOrderableDbInstanceOptionsRequest;
import software.amazon.awssdk.services.rds.model.DeleteDbParameterGroupRequest;
import software.amazon.awssdk.services.secretsmanager.SecretsManagerClient;
import software.amazon.awssdk.services.secretsmanager.model.GetSecretValueRequest;
import software.amazon.awssdk.services.secretsmanager.model.GetSecretValueResponse;
import java.util.ArrayList;
import java.util.List;

/**
 * Before running this Java (v2) code example, set up your development
 * environment, including your credentials.
 *

```



```

* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*
* This example requires an AWS Secrets Manager secret that contains the
* database credentials. If you do not create a
* secret, this example will not work. For details, see:
*
* https://docs.aws.amazon.com/secretsmanager/latest/userguide/integrating\_how-
services-use-secrets\_RS.html
*
* This Java example performs these tasks:
*
* 1. Returns a list of the available DB engines.
* 2. Selects an engine family and create a custom DB parameter group.
* 3. Gets the parameter groups.
* 4. Gets parameters in the group.
* 5. Modifies the auto_increment_offset parameter.
* 6. Gets and displays the updated parameters.
* 7. Gets a list of allowed engine versions.
* 8. Gets a list of micro instance classes available for the selected engine.
* 9. Creates an RDS database instance that contains a MySQL database and uses
* the parameter group.
* 10. Waits for the DB instance to be ready and prints out the connection
* endpoint value.
* 11. Creates a snapshot of the DB instance.
* 12. Waits for an RDS DB snapshot to be ready.
* 13. Deletes the RDS DB instance.
* 14. Deletes the parameter group.
*/
public class RDSScenario {
    public static long sleepTime = 20;
    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    public static void main(String[] args) throws InterruptedException {
        final String usage = ""

                Usage:
                <dbGroupName> <dbParameterGroupFamily> <dbInstanceIdentifier>
<dbName> <dbSnapshotIdentifier> <secretName>

                Where:
                dbGroupName - The database group name.\s

```

```

        dbParameterGroupFamily - The database parameter group name (for
example, mysql8.0).
        dbInstanceIdentifier - The database instance identifier\s
        dbName - The database name.\s
        dbSnapshotIdentifier - The snapshot identifier.\s
        secretName - The name of the AWS Secrets Manager secret that
contains the database credentials"
        """;

    if (args.length != 6) {
        System.out.println(usage);
        System.exit(1);
    }

    String dbGroupName = args[0];
    String dbParameterGroupFamily = args[1];
    String dbInstanceIdentifier = args[2];
    String dbName = args[3];
    String dbSnapshotIdentifier = args[4];
    String secretName = args[5];

    Gson gson = new Gson();
    User user = gson.fromJson(String.valueOf(getSecretValues(secretName)),
User.class);
    String masterUsername = user.getUsername();
    String masterUserPassword = user.getPassword();

    Region region = Region.US_WEST_2;
    RdsClient rdsClient = RdsClient.builder()
        .region(region)
        .build();
    System.out.println(DASHES);
    System.out.println("Welcome to the Amazon RDS example scenario.");
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("1. Return a list of the available DB engines");
    describeDBEngines(rdsClient);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("2. Create a custom parameter group");
    createDBParameterGroup(rdsClient, dbGroupName, dbParameterGroupFamily);
    System.out.println(DASHES);

```

```
System.out.println(DASHES);
System.out.println("3. Get the parameter group");
describeDbParameterGroups(rdsClient, dbGroupName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("4. Get the parameters in the group");
describeDbParameters(rdsClient, dbGroupName, 0);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("5. Modify the auto_increment_offset parameter");
modifyDBParas(rdsClient, dbGroupName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("6. Display the updated value");
describeDbParameters(rdsClient, dbGroupName, -1);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("7. Get a list of allowed engine versions");
getAllowedEngines(rdsClient, dbParameterGroupFamily);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("8. Get a list of micro instance classes available for
the selected engine");
getMicroInstances(rdsClient);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println(
    "9. Create an RDS database instance that contains a MySQL database
and uses the parameter group");
String dbARN = createDatabaseInstance(rdsClient, dbGroupName,
dbInstanceIdentifier, dbName, masterUsername,
    masterUserPassword);
System.out.println("The ARN of the new database is " + dbARN);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("10. Wait for DB instance to be ready");
```

```
    waitForInstanceReady(rdsClient, dbInstanceIdentifier);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("11. Create a snapshot of the DB instance");
    createSnapshot(rdsClient, dbInstanceIdentifier, dbSnapshotIdentifier);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("12. Wait for DB snapshot to be ready");
    waitForSnapshotReady(rdsClient, dbInstanceIdentifier, dbSnapshotIdentifier);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("13. Delete the DB instance");
    deleteDatabaseInstance(rdsClient, dbInstanceIdentifier);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("14. Delete the parameter group");
    deleteParaGroup(rdsClient, dbGroupName, dbARN);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("The Scenario has successfully completed.");
    System.out.println(DASHES);

    rdsClient.close();
}

private static SecretsManagerClient getSecretClient() {
    Region region = Region.US_WEST_2;
    return SecretsManagerClient.builder()
        .region(region)
        .credentialsProvider(EnvironmentVariableCredentialsProvider.create())
        .build();
}

public static String getSecretValues(String secretName) {
    SecretsManagerClient secretClient = getSecretClient();
    GetSecretValueRequest valueRequest = GetSecretValueRequest.builder()
        .secretId(secretName)
        .build();
```

```
        GetSecretValueResponse valueResponse =
secretClient.getSecretValue(valueRequest);
        return valueResponse.secretString();
    }

    // Delete the parameter group after database has been deleted.
    // An exception is thrown if you attempt to delete the para group while database
    // exists.
    public static void deleteParaGroup(RdsClient rdsClient, String dbGroupName,
String dbARN)
        throws InterruptedException {
    try {
        boolean isDataDel = false;
        boolean didFind;
        String instanceARN;

        // Make sure that the database has been deleted.
        while (!isDataDel) {
            DescribeDbInstancesResponse response =
rdsClient.describeDBInstances();
            List<DBInstance> instanceList = response.dbInstances();
            int listSize = instanceList.size();
            didFind = false;
            int index = 1;
            for (DBInstance instance : instanceList) {
                instanceARN = instance.dbInstanceArn();
                if (instanceARN.compareTo(dbARN) == 0) {
                    System.out.println(dbARN + " still exists");
                    didFind = true;
                }
            }
            if ((index == listSize) && (!didFind)) {
                // Went through the entire list and did not find the
database ARN.
                isDataDel = true;
            }
            Thread.sleep(sleepTime * 1000);
            index++;
        }
    }

    // Delete the para group.
    DeleteDbParameterGroupRequest parameterGroupRequest =
DeleteDbParameterGroupRequest.builder()
```

```
        .dbParameterGroupName(dbGroupName)
        .build();

        rdsClient.deleteDBParameterGroup(parameterGroupRequest);
        System.out.println(dbGroupName + " was deleted.");

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}

// Delete the DB instance.
public static void deleteDatabaseInstance(RdsClient rdsClient, String
dbInstanceIdentifier) {
    try {
        DeleteDbInstanceRequest deleteDbInstanceRequest =
DeleteDbInstanceRequest.builder()
        .dbInstanceIdentifier(dbInstanceIdentifier)
        .deleteAutomatedBackups(true)
        .skipFinalSnapshot(true)
        .build();

        DeleteDbInstanceResponse response =
rdsClient.deleteDBInstance(deleteDbInstanceRequest);
        System.out.print("The status of the database is " +
response.dbInstance().dbInstanceStatus());

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}

// Waits until the snapshot instance is available.
public static void waitForSnapshotReady(RdsClient rdsClient, String
dbInstanceIdentifier,
    String dbSnapshotIdentifier) {
    try {
        boolean snapshotReady = false;
        String snapshotReadyStr;
        System.out.println("Waiting for the snapshot to become available.");
```

```
        DescribeDbSnapshotsRequest snapshotsRequest =
DescribeDbSnapshotsRequest.builder()
        .dbSnapshotIdentifier(dbSnapshotIdentifier)
        .dbInstanceIdentifier(dbInstanceIdentifier)
        .build();

        while (!snapshotReady) {
            DescribeDbSnapshotsResponse response =
rdsClient.describeDBSnapshots(snapshotsRequest);
            List<DBSnapshot> snapshotList = response.dbSnapshots();
            for (DBSnapshot snapshot : snapshotList) {
                snapshotReadyStr = snapshot.status();
                if (snapshotReadyStr.contains("available")) {
                    snapshotReady = true;
                } else {
                    System.out.print(".");
                    Thread.sleep(sleepTime * 1000);
                }
            }
        }

        System.out.println("The Snapshot is available!");
    } catch (RdsException | InterruptedException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}

// Create an Amazon RDS snapshot.
public static void createSnapshot(RdsClient rdsClient, String
dbInstanceIdentifier, String dbSnapshotIdentifier) {
    try {
        CreateDbSnapshotRequest snapshotRequest =
CreateDbSnapshotRequest.builder()
        .dbInstanceIdentifier(dbInstanceIdentifier)
        .dbSnapshotIdentifier(dbSnapshotIdentifier)
        .build();

        CreateDbSnapshotResponse response =
rdsClient.createDBSnapshot(snapshotRequest);
        System.out.println("The Snapshot id is " +
response.dbSnapshot().dbiResourceId());

    } catch (RdsException e) {
```

```
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}

// Waits until the database instance is available.
public static void waitForInstanceReady(RdsClient rdsClient, String
dbInstanceIdentifier) {
    boolean instanceReady = false;
    String instanceReadyStr;
    System.out.println("Waiting for instance to become available.");
    try {
        DescribeDbInstancesRequest instanceRequest =
DescribeDbInstancesRequest.builder()
            .dbInstanceIdentifier(dbInstanceIdentifier)
            .build();

        String endpoint = "";
        while (!instanceReady) {
            DescribeDbInstancesResponse response =
rdsClient.describeDBInstances(instanceRequest);
            List<DBInstance> instanceList = response.dbInstances();
            for (DBInstance instance : instanceList) {
                instanceReadyStr = instance.dbInstanceStatus();
                if (instanceReadyStr.contains("available")) {
                    endpoint = instance.endpoint().address();
                    instanceReady = true;
                } else {
                    System.out.print(".");
                    Thread.sleep(sleepTime * 1000);
                }
            }
        }
        System.out.println("Database instance is available! The connection
endpoint is " + endpoint);

    } catch (RdsException | InterruptedException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

// Create a database instance and return the ARN of the database.
public static String createDatabaseInstance(RdsClient rdsClient,
```



```
        String dbGroupName,
        String dbInstanceIdentifier,
        String dbName,
        String masterUsername,
        String masterUserPassword) {

    try {
        CreateDbInstanceRequest instanceRequest =
CreateDbInstanceRequest.builder()
            .dbInstanceIdentifier(dbInstanceIdentifier)
            .allocatedStorage(100)
            .dbName(dbName)
            .dbParameterGroupName(dbGroupName)
            .engine("mysql")
            .dbInstanceClass("db.m4.large")
            .engineVersion("8.0")
            .storageType("standard")
            .masterUsername(masterUsername)
            .masterUserPassword(masterUserPassword)
            .build();

        CreateDbInstanceResponse response =
rdsClient.createDBInstance(instanceRequest);
        System.out.print("The status is " +
response.dbInstance().dbInstanceStatus());
        return response.dbInstance().dbInstanceArn();

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }

    return "";
}

// Get a list of micro instances.
public static void getMicroInstances(RdsClient rdsClient) {
    try {
        DescribeOrderableDbInstanceOptionsRequest dbInstanceOptionsRequest =
DescribeOrderableDbInstanceOptionsRequest
            .builder()
            .engine("mysql")
            .build();
```

```
        DescribeOrderableDbInstanceOptionsResponse response = rdsClient
            .describeOrderableDBInstanceOptions(dbInstanceOptionsRequest);
        List<OrderableDBInstanceOption> orderableDBInstances =
response.orderableDBInstanceOptions();
        for (OrderableDBInstanceOption dbInstanceOption : orderableDBInstances)
    {
            System.out.println("The engine version is " +
dbInstanceOption.engineVersion());
            System.out.println("The engine description is " +
dbInstanceOption.engine());
        }

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}

// Get a list of allowed engine versions.
public static void getAllowedEngines(RdsClient rdsClient, String
dbParameterGroupFamily) {
    try {
        DescribeDbEngineVersionsRequest versionsRequest =
DescribeDbEngineVersionsRequest.builder()
            .dbParameterGroupFamily(dbParameterGroupFamily)
            .engine("mysql")
            .build();

        DescribeDbEngineVersionsResponse response =
rdsClient.describeDBEngineVersions(versionsRequest);
        List<DBEngineVersion> dbEngines = response.dbEngineVersions();
        for (DBEngineVersion dbEngine : dbEngines) {
            System.out.println("The engine version is " +
dbEngine.engineVersion());
            System.out.println("The engine description is " +
dbEngine.dbEngineDescription());
        }

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}
```

```
// Modify auto_increment_offset and auto_increment_increment parameters.
public static void modifyDBParas(RdsClient rdsClient, String dbGroupName) {
    try {
        Parameter parameter1 = Parameter.builder()
            .parameterName("auto_increment_offset")
            .applyMethod("immediate")
            .parameterValue("5")
            .build();

        List<Parameter> paraList = new ArrayList<>();
        paraList.add(parameter1);
        ModifyDbParameterGroupRequest groupRequest =
ModifyDbParameterGroupRequest.builder()
            .dbParameterGroupName(dbGroupName)
            .parameters(paraList)
            .build();

        ModifyDbParameterGroupResponse response =
rdsClient.modifyDBParameterGroup(groupRequest);
        System.out.println("The parameter group " +
response.dbParameterGroupName() + " was successfully modified");

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}

// Retrieve parameters in the group.
public static void describeDbParameters(RdsClient rdsClient, String dbGroupName,
int flag) {
    try {
        DescribeDbParametersRequest dbParameterGroupsRequest;
        if (flag == 0) {
            dbParameterGroupsRequest = DescribeDbParametersRequest.builder()
                .dbParameterGroupName(dbGroupName)
                .build();
        } else {
            dbParameterGroupsRequest = DescribeDbParametersRequest.builder()
                .dbParameterGroupName(dbGroupName)
                .source("user")
                .build();
        }
    }
}
```

```

        DescribeDbParametersResponse response =
rdsClient.describeDBParameters(dbParameterGroupsRequest);
        List<Parameter> dbParameters = response.parameters();
        String paraName;
        for (Parameter para : dbParameters) {
            // Only print out information about either auto_increment_offset or
            // auto_increment_increment.
            paraName = para.parameterName();
            if ((paraName.compareTo("auto_increment_offset") == 0)
                || (paraName.compareTo("auto_increment_increment ") == 0)) {
                System.out.println("*** The parameter name is " + paraName);
                System.out.println("*** The parameter value is " +
para.parameterValue());
                System.out.println("*** The parameter data type is " +
para.dataType());
                System.out.println("*** The parameter description is " +
para.description());
                System.out.println("*** The parameter allowed values is " +
para.allowedValues());
            }
        }

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}

public static void describeDbParameterGroups(RdsClient rdsClient, String
dbGroupName) {
    try {
        DescribeDbParameterGroupsRequest groupsRequest =
DescribeDbParameterGroupsRequest.builder()
            .dbParameterGroupName(dbGroupName)
            .maxRecords(20)
            .build();

        DescribeDbParameterGroupsResponse response =
rdsClient.describeDBParameterGroups(groupsRequest);
        List<DBParameterGroup> groups = response.dbParameterGroups();
        for (DBParameterGroup group : groups) {
            System.out.println("The group name is " +
group.dbParameterGroupName());
        }
    }
}

```

```
        System.out.println("The group description is " +
group.description());
    }

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}

public static void createDBParameterGroup(RdsClient rdsClient, String
dbGroupName, String dbParameterGroupFamily) {
    try {
        CreateDbParameterGroupRequest groupRequest =
CreateDbParameterGroupRequest.builder()
            .dbParameterGroupName(dbGroupName)
            .dbParameterGroupFamily(dbParameterGroupFamily)
            .description("Created by using the AWS SDK for Java")
            .build();

        CreateDbParameterGroupResponse response =
rdsClient.createDBParameterGroup(groupRequest);
        System.out.println("The group name is " +
response.dbParameterGroup().dbParameterGroupName());

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}

public static void describeDBEngines(RdsClient rdsClient) {
    try {
        DescribeDbEngineVersionsRequest engineVersionsRequest =
DescribeDbEngineVersionsRequest.builder()
            .defaultOnly(true)
            .engine("mysql")
            .maxRecords(20)
            .build();

        DescribeDbEngineVersionsResponse response =
rdsClient.describeDBEngineVersions(engineVersionsRequest);
        List<DBEngineVersion> engines = response.dbEngineVersions();
    }
}
```

```
// Get all DBEngineVersion objects.
for (DBEngineVersion engineOb : engines) {
    System.out.println("The name of the DB parameter group family for
the database engine is "
        + engineOb.dbParameterGroupFamily());
    System.out.println("The name of the database engine " +
engineOb.engine());
    System.out.println("The version number of the database engine " +
engineOb.engineVersion());
}

} catch (RdsException e) {
    System.out.println(e.getLocalizedMessage());
    System.exit(1);
}
}
```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella Documentazione di riferimento delle API AWS SDK for Java 2.x .
  - [CreateDBInstance](#)
  - [Creato B ParameterGroup](#)
  - [CreateDBSnapshot](#)
  - [DeleteDBInstance](#)
  - [Eliminare DB ParameterGroup](#)
  - [Descritto B EngineVersions](#)
  - [DescribeDBInstances](#)
  - [Descritto B ParameterGroups](#)
  - [DescribeDBParameters](#)
  - [DescribeDBSnapshots](#)
  - [DescribeOrderableDB InstanceOptions](#)
  - [Modifica DB ParameterGroup](#)

## Esempi di Amazon Redshift con SDK for Java 2.x

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando Amazon Redshift. AWS SDK for Java 2.x

Le operazioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le operazioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati e negli esempi tra servizi.

Scenari: esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio.

Ogni esempio include un collegamento a GitHub, dove puoi trovare istruzioni su come configurare ed eseguire il codice nel contesto.

Nozioni di base

Salve Amazon Redshift

I seguenti esempi di codice mostrano come iniziare a usare Amazon Redshift.

SDK per Java 2.x

### Note

C'è altro su [GitHub](#) Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.redshift.RedshiftClient;
import software.amazon.awssdk.services.redshift.paginators.DescribeClustersIterable;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class HelloRedshift {
```

```
public static void main(String[] args) {
    Region region = Region.US_EAST_1;
    RedshiftClient redshiftClient = RedshiftClient.builder()
        .region(region)
        .build();

    listClustersPaginator(redshiftClient);
}

public static void listClustersPaginator(RedshiftClient redshiftClient) {
    DescribeClustersIterable clustersIterable =
redshiftClient.describeClustersPaginator();
    clustersIterable.stream()
        .flatMap(r -> r.clusters().stream())
        .forEach(cluster -> System.out
            .println(" Cluster identifier: " + cluster.clusterIdentifier() + "
status = " + cluster.clusterStatus()));
}
}
```

- Per i dettagli sull'API, consulta [DescribeClusters](#) in AWS SDK for Java 2.x API Reference.

## Argomenti

- [Azioni](#)
- [Scenari](#)

## Azioni

### CreateCluster

Il seguente esempio di codice mostra come utilizzare `CreateCluster`

SDK per Java 2.x

#### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).



## Crea il cluster .

```
public static void createCluster(RedshiftClient redshiftClient, String
clusterId, String masterUsername,
                                String masterUserPassword) {
    try {
        CreateClusterRequest clusterRequest = CreateClusterRequest.builder()
            .clusterIdentifier(clusterId)
            .masterUsername(masterUsername)
            .masterUserPassword(masterUserPassword)
            .nodeType("ra3.4xlarge")
            .publiclyAccessible(true)
            .numberOfNodes(2)
            .build();

        CreateClusterResponse clusterResponse =
redshiftClient.createCluster(clusterRequest);
        System.out.println("Created cluster " +
clusterResponse.cluster().clusterIdentifier());

    } catch (RedshiftException e) {

        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- Per i dettagli sull'API, consulta la [CreateCluster](#) sezione AWS SDK for Java 2.x API Reference.

## CreateTable

Il seguente esempio di codice mostra come utilizzare `CreateTable`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public static void createTable(RedshiftDataClient redshiftDataClient, String
clusterId, String databaseName, String userName) {
    try {
        ExecuteStatementRequest createTableRequest =
ExecuteStatementRequest.builder()
        .clusterIdentifier(clusterId)
        .dbUser(userName)
        .database(databaseName)
        .sql("CREATE TABLE Movies ("
            + "id INT PRIMARY KEY, "
            + "title VARCHAR(100), "
            + "year INT)")
        .build();

        redshiftDataClient.executeStatement(createTableRequest);
        System.out.println("Table created: Movies");

    } catch (RedshiftDataException e) {
        System.err.println("Error creating table: " + e.getMessage());
        System.exit(1);
    }
}
```

- Per i dettagli sull'API, consulta la [CreateTable](#) sezione AWS SDK for Java 2.x API Reference.

## DeleteCluster

Il seguente esempio di codice mostra come utilizzare `DeleteCluster`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Elimina il cluster.

```
public static void deleteRedshiftCluster(RedshiftClient redshiftClient, String
clusterId) {
    try {
        DeleteClusterRequest deleteClusterRequest =
DeleteClusterRequest.builder()
        .clusterIdentifier(clusterId)
        .skipFinalClusterSnapshot(true)
        .build();

        DeleteClusterResponse response =
redshiftClient.deleteCluster(deleteClusterRequest);
        System.out.println("The status is " +
response.cluster().clusterStatus());

    } catch (RedshiftException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- Per i dettagli sull'API, consulta la [DeleteCluster](#) sezione AWS SDK for Java 2.x API Reference.

## DescribeClusters

Il seguente esempio di codice mostra come utilizzare `DescribeClusters`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Descrivi il cluster.

```
public static void waitForClusterReady(RedshiftClient redshiftClient, String
clusterId) {
    boolean clusterReady = false;
    String clusterReadyStr;
```

```
        System.out.println("Waiting for cluster to become available. This may take a
few mins.");
        try {
            DescribeClustersRequest clustersRequest =
DescribeClustersRequest.builder()
                .clusterIdentifier(clusterId)
                .build();
            long startTime = System.currentTimeMillis();

            // Loop until the cluster is ready.
            while (!clusterReady) {
                DescribeClustersResponse clusterResponse =
redshiftClient.describeClusters(clustersRequest);
                List<Cluster> clusterList = clusterResponse.clusters();
                for (Cluster cluster : clusterList) {
                    clusterReadyStr = cluster.clusterStatus();
                    if (clusterReadyStr.contains("available"))
                        clusterReady = true;
                    else {
                        long elapsedTimeMillis = System.currentTimeMillis() -
startTime;

                        long elapsedSeconds = elapsedTimeMillis / 1000;
                        long minutes = elapsedSeconds / 60;
                        long seconds = elapsedSeconds % 60;

                        System.out.printf("Elapsed Time: %02d:%02d - Waiting for
cluster... %n", minutes, seconds);
                        TimeUnit.SECONDS.sleep(5);
                    }
                }
            }

            long elapsedTimeMillis = System.currentTimeMillis() - startTime;
            long elapsedSeconds = elapsedTimeMillis / 1000;
            long minutes = elapsedSeconds / 60;
            long seconds = elapsedSeconds % 60;

            System.out.println(String.format("Cluster is available! Total Elapsed
Time: %02d:%02d", minutes, seconds));

        } catch (RedshiftException | InterruptedException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

```
}
```

- Per i dettagli sull'API, [DescribeClusters](#) consulta AWS SDK for Java 2.x API Reference.

## DescribeStatement

Il seguente esempio di codice mostra come utilizzare `DescribeStatement`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public static void checkStatement(RedshiftDataClient redshiftDataClient, String
sqlId) {
    try {
        DescribeStatementRequest statementRequest =
DescribeStatementRequest.builder()
            .id(sqlId)
            .build();

        String status;
        while (true) {
            DescribeStatementResponse response =
redshiftDataClient.describeStatement(statementRequest);
            status = response.statusAsString();
            System.out.println("..." + status);

            if (status.compareTo("FAILED") == 0 ) {
                System.out.println("The Query Failed. Ending program");
                System.exit(1);
            } else if (status.compareTo("FINISHED") == 0) {
                break;
            }
            TimeUnit.SECONDS.sleep(1);
        }
    }
}
```

```
        System.out.println("The statement is finished!");
    } catch (RedshiftDataException | InterruptedException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- Per i dettagli sull'API, consulta la [DescribeStatement](#) sezione AWS SDK for Java 2.x API Reference.

## GetStatementResult

Il seguente esempio di codice mostra come utilizzare `GetStatementResult`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Controlla il risultato della dichiarazione.

```
public static void getResults(RedshiftDataClient redshiftDataClient, String
statementId) {
    try {
        GetStatementResultRequest resultRequest =
GetStatementResultRequest.builder()
            .id(statementId)
            .build();

        // Extract and print the field values using streams.
        GetStatementResultResponse response =
redshiftDataClient.getStatementResult(resultRequest);
        response.records().stream()
            .flatMap(List::stream)
            .map(Field::stringValue)
            .filter(value -> value != null)
```

```

        .forEach(value -> System.out.println("The Movie title field is " +
value));

    } catch (RedshiftDataException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

```

- Per i dettagli sull'API, consulta la [GetStatementResult](#) sezione AWS SDK for Java 2.x API Reference.

## Insert

Il seguente esempio di codice mostra come utilizzare `Insert`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```

public static void popTable(RedshiftDataClient redshiftDataClient, String
clusterId, String databaseName, String userName, String fileName, int number)
throws IOException {
    JsonParser parser = new JsonFactory().createParser(new File(fileName));
    com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
    Iterator<JsonNode> iter = rootNode.iterator();
    ObjectNode currentNode;
    int t = 0;
    while (iter.hasNext()) {
        if (t == number)
            break;
        currentNode = (ObjectNode) iter.next();
        int year = currentNode.get("year").asInt();
        String title = currentNode.get("title").asText();

```

```
// Use SqlParameter to avoid SQL injection.
List<SqlParameter> parameterList = new ArrayList<>();
String sqlStatement = "INSERT INTO Movies
VALUES( :id , :title, :year)";

// Create the parameters.
SqlParameter idParam = SqlParameter.builder()
    .name("id")
    .value(String.valueOf(t))
    .build();

SqlParameter titleParam= SqlParameter.builder()
    .name("title")
    .value(title)
    .build();

SqlParameter yearParam = SqlParameter.builder()
    .name("year")
    .value(String.valueOf(year))
    .build();
parameterList.add(idParam);
parameterList.add(titleParam);
parameterList.add(yearParam);

try {
    ExecuteStatementRequest insertStatementRequest =
ExecuteStatementRequest.builder()
    .clusterIdentifier(clusterId)
    .sql(sqlStatement)
    .database(databaseName)
    .dbUser(userName)
    .parameters(parameterList)
    .build();

    redshiftDataClient.executeStatement(insertStatementRequest);
    System.out.println("Inserted: " + title + " (" + year + ")");
    t++;

} catch (RedshiftDataException e) {
    System.err.println("Error inserting data: " + e.getMessage());
    System.exit(1);
}
}
System.out.println(t + " records were added to the Movies table. ");
```



```
}
```

- Per i dettagli sull'API, consulta [Insert](#) in AWS SDK for Java 2.x API Reference.

## ModifyCluster

Il seguente esempio di codice mostra come utilizzare `ModifyCluster`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Modifica un cluster.

```
public static void modifyCluster(RedshiftClient redshiftClient, String
clusterId) {
    try {
        ModifyClusterRequest modifyClusterRequest =
ModifyClusterRequest.builder()
            .clusterIdentifier(clusterId)
            .preferredMaintenanceWindow("wed:07:30-wed:08:00")
            .build();

        ModifyClusterResponse clusterResponse =
redshiftClient.modifyCluster(modifyClusterRequest);
        System.out.println("The modified cluster was successfully modified and
has "
            + clusterResponse.cluster().preferredMaintenanceWindow() + " as the
maintenance window");

    } catch (RedshiftException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- Per i dettagli sull'API, consulta la [ModifyCluster](#) sezione AWS SDK for Java 2.x API Reference.

## Query

Il seguente esempio di codice mostra come utilizzare `Query`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Interroga una tabella.

```
public static String queryMoviesByYear(RedshiftDataClient redshiftDataClient,
                                       String database,
                                       String dbUser,
                                       int year,
                                       String clusterId) {

    try {
        String sqlStatement = " SELECT * FROM Movies WHERE year = :year";
        SqlParameter yearParam= SqlParameter.builder()
            .name("year")
            .value(String.valueOf(year))
            .build();

        ExecuteStatementRequest statementRequest =
ExecuteStatementRequest.builder()
            .clusterIdentifier(clusterId)
            .database(database)
            .dbUser(dbUser)
            .parameters(yearParam)
            .sql(sqlStatement)
            .build();

        ExecuteStatementResponse response =
redshiftDataClient.executeStatement(statementRequest);
        return response.id();
    }
}
```

```
    } catch (RedshiftDataException e) {  
        System.err.println(e.getMessage());  
        System.exit(1);  
    }  
    return "";  
}
```

- Per ulteriori informazioni sulle API, consulta [Query](#) nella Documentazione di riferimento delle API AWS SDK for Java 2.x .

## Scenari

### Inizia a usare Amazon Redshift

Il seguente esempio di codice mostra come lavorare con tabelle, elementi e query di Amazon Redshift.

### SDK per Java 2.x

#### Note

C'è di più su [GitHub](#) Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import com.fasterxml.jackson.core.JsonFactory;  
import com.fasterxml.jackson.databind.JsonNode;  
import com.fasterxml.jackson.databind.ObjectMapper;  
import com.fasterxml.jackson.databind.node.ObjectNode;  
import software.amazon.awssdk.regions.Region;  
import software.amazon.awssdk.services.redshift.RedshiftClient;  
import software.amazon.awssdk.services.redshift.model.Cluster;  
import software.amazon.awssdk.services.redshift.model.CreateClusterRequest;  
import software.amazon.awssdk.services.redshift.model.CreateClusterResponse;  
import software.amazon.awssdk.services.redshift.model.DeleteClusterRequest;  
import software.amazon.awssdk.services.redshift.model.DeleteClusterResponse;  
import software.amazon.awssdk.services.redshift.model.DescribeClustersRequest;  
import software.amazon.awssdk.services.redshift.model.DescribeClustersResponse;  
import software.amazon.awssdk.services.redshift.model.ModifyClusterRequest;  
import software.amazon.awssdk.services.redshift.model.ModifyClusterResponse;
```

```
import software.amazon.awssdk.services.redshift.model.RedshiftException;
import software.amazon.awssdk.services.redshiftdata.RedshiftDataClient;
import software.amazon.awssdk.services.redshiftdata.model.DescribeStatementRequest;
import software.amazon.awssdk.services.redshiftdata.model.DescribeStatementResponse;
import software.amazon.awssdk.services.redshiftdata.model.ExecuteStatementRequest;
import software.amazon.awssdk.services.redshiftdata.model.ExecuteStatementResponse;
import software.amazon.awssdk.services.redshiftdata.model.Field;
import software.amazon.awssdk.services.redshiftdata.model.GetStatementResultRequest;
import
    software.amazon.awssdk.services.redshiftdata.model.GetStatementResultResponse;
import software.amazon.awssdk.services.redshiftdata.model.ListDatabasesRequest;
import software.amazon.awssdk.services.redshiftdata.model.RedshiftDataException;
import software.amazon.awssdk.services.redshiftdata.model.SqlParameter;
import
    software.amazon.awssdk.services.redshiftdata.paginators.ListDatabasesIterable;
import com.fasterxml.jackson.core.JsonParser;
import java.io.File;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;
import java.util.Scanner;
import java.util.concurrent.TimeUnit;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * This Java example performs these tasks:
 *
 * 1. Prompts the user for a unique cluster ID or use the default value.
 * 2. Creates a Redshift cluster with the specified or default cluster Id value.
 * 3. Waits until the Redshift cluster is available for use.
 * 4. Lists all databases using a pagination API call.
 * 5. Creates a table named "Movies" with fields ID, title, and year.
 * 6. Inserts a specified number of records into the "Movies" table by reading the
    Movies JSON file.
 * 7. Prompts the user for a movie release year.
 * 8. Runs a SQL query to retrieve movies released in the specified year.
```

- \* 9. Modifies the Redshift cluster.
- \* 10. Prompts the user for confirmation to delete the Redshift cluster.
- \* 11. If confirmed, deletes the specified Redshift cluster.

```

*/

public class RedshiftScenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");
    public static void main(String[] args) throws Exception {
        final String usage = ""

            Usage:
                <jsonFilePath>\s

            Where:
                jsonFilePath - The path to the Movies JSON file (you can locate that
file in ../../../../resources/sample_files/movies.json)
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String jsonFilePath = args[0];
        String userName;
        String userPassword;
        String databaseName = "dev" ;
        Scanner scanner = new Scanner(System.in);

        Region region = Region.US_EAST_1;
        RedshiftClient redshiftClient = RedshiftClient.builder()
            .region(region)
            .build();

        RedshiftDataClient redshiftDataClient = RedshiftDataClient.builder()
            .region(region)
            .build();

        System.out.println(DASHES);
        System.out.println("Welcome to the Amazon Redshift SDK Getting Started
scenario.");
        System.out.println("""
            This Java program demonstrates how to interact with Amazon Redshift by using
the AWS SDK for Java (v2).\s

```

Amazon Redshift is a fully managed, petabyte-scale data warehouse service hosted in the cloud.

The program's primary functionalities include cluster creation, verification of cluster readiness,\s

list databases, table creation, data population within the table, and execution of SQL statements.

Furthermore, it demonstrates the process of querying data from the Movie table.\s

Upon completion of the program, all AWS resources are cleaned up.  
 """);

```

System.out.println("Lets get started...");
System.out.println("Please enter your user name (default is awsuser)");
String user = scanner.nextLine();
userName = user.isEmpty() ? "awsuser" : user;
System.out.println(DASHES);
System.out.println("Please enter your user password (default is
AwsUser1000)");
String userpass = scanner.nextLine();
userPassword = userpass.isEmpty() ? "AwsUser1000" : userpass;
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("A Redshift cluster refers to the collection of computing
resources and storage that work together to process and analyze large volumes of
data.");
System.out.println("Enter a cluster id value (default is redshift-cluster-
movies): ");
String userClusterId = scanner.nextLine();
String clusterId = userClusterId.isEmpty() ? "redshift-cluster-movies" :
userClusterId;
createCluster(redshiftClient, clusterId, userName, userPassword);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("Wait until "+clusterId +" is available.");
System.out.print("Press Enter to continue...");
scanner.nextLine();
waitForClusterReady(redshiftClient, clusterId);
System.out.println(DASHES);

System.out.println(DASHES);

```

```
String databaseInfo = ""
    When you created $clusteridD, the dev database is created by default and
used in this scenario.\s
```

```
    To create a custom database, you need to have a CREATEDB privilege.\s
```

```
    For more information, see the documentation here: https://docs.aws.amazon.com/redshift/latest/dg/r\_CREATE\_DATABASE.html.
```

```
    """.replace("$clusteridD", clusterId);
```

```
System.out.println(databaseInfo);
System.out.print("Press Enter to continue...");
scanner.nextLine();
System.out.println(DASHES);
```

```
System.out.println(DASHES);
System.out.println("List databases in "+clusterId);
System.out.print("Press Enter to continue...");
scanner.nextLine();
listAllDatabases(redshiftDataClient, clusterId, userName, databaseName);
System.out.println(DASHES);
```

```
System.out.println(DASHES);
System.out.println("Now you will create a table named Movies.");
System.out.print("Press Enter to continue...");
scanner.nextLine();
createTable(redshiftDataClient, clusterId, databaseName, userName);
System.out.println(DASHES);
```

```
System.out.println(DASHES);
System.out.println("Populate the Movies table using the Movies.json file.");
System.out.println("Specify the number of records you would like to add to
the Movies Table.");
System.out.println("Please enter a value between 50 and 200.");
int numRecords;
do {
    System.out.print("Enter a value: ");
    while (!scanner.hasNextInt()) {
        System.out.println("Invalid input. Please enter a value between 50
and 200.");
        System.out.print("Enter a year: ");
        scanner.next();
    }
    numRecords = scanner.nextInt();
} while (numRecords < 50 || numRecords > 200);
```

```
        popTable(redshiftDataClient, clusterId, databaseName, userName,
jsonFilePath, numRecords);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Query the Movies table by year. Enter a value between
2012-2014.");
        int movieYear;
        do {
            System.out.print("Enter a year: ");
            while (!scanner.hasNextInt()) {
                System.out.println("Invalid input. Please enter a valid year between
2012 and 2014.");
                System.out.print("Enter a year: ");
                scanner.next();
            }
            movieYear = scanner.nextInt();
            scanner.nextLine();
        } while (movieYear < 2012 || movieYear > 2014);

        String id = queryMoviesByYear(redshiftDataClient, databaseName, userName,
movieYear, clusterId);
        System.out.println("The identifier of the statement is " + id);
        checkStatement(redshiftDataClient, id);
        getResults(redshiftDataClient, id);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Now you will modify the Redshift cluster.");
        System.out.print("Press Enter to continue...");
        scanner.nextLine();
        modifyCluster(redshiftClient, clusterId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Would you like to delete the Amazon Redshift cluster?
(y/n)");
        String delAns = scanner.nextLine().trim();
        if (delAns.equalsIgnoreCase("y")) {
            System.out.println("You selected to delete " +clusterId);
            System.out.print("Press Enter to continue...");
            scanner.nextLine();
            deleteRedshiftCluster(redshiftClient, clusterId);
        } else {
```



```
        System.out.println("The "+clusterId +" was not deleted");
    }
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("This concludes the Amazon Redshift SDK Getting Started
scenario.");
    System.out.println(DASHES);
}

public static void listAllDatabases(RedshiftDataClient redshiftDataClient,
String clusterId, String dbUser, String database) {
    try {
        ListDatabasesRequest databasesRequest = ListDatabasesRequest.builder()
            .clusterIdentifier(clusterId)
            .dbUser(dbUser)
            .database(database)
            .build();

        ListDatabasesIterable listDatabasesIterable =
redshiftDataClient.listDatabasesPaginator(databasesRequest);
        listDatabasesIterable.stream()
            .flatMap(r -> r.databases().stream())
            .forEach(db -> System.out
                .println("The database name is : " + db));

    } catch (RedshiftDataException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void deleteRedshiftCluster(RedshiftClient redshiftClient, String
clusterId) {
    try {
        DeleteClusterRequest deleteClusterRequest =
DeleteClusterRequest.builder()
            .clusterIdentifier(clusterId)
            .skipFinalClusterSnapshot(true)
            .build();

        DeleteClusterResponse response =
redshiftClient.deleteCluster(deleteClusterRequest);
    }
}
```

```
        System.out.println("The status is " +
response.cluster().clusterStatus());

    } catch (RedshiftException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void popTable(RedshiftDataClient redshiftDataClient, String
clusterId, String databaseName, String userName, String fileName, int number)
throws IOException {
    JsonParser parser = new JsonFactory().createParser(new File(fileName));
    com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
    Iterator<JsonNode> iter = rootNode.iterator();
    ObjectNode currentNode;
    int t = 0;
    while (iter.hasNext()) {
        if (t == number)
            break;
        currentNode = (ObjectNode) iter.next();
        int year = currentNode.get("year").asInt();
        String title = currentNode.get("title").asText();

        // Use SqlParameter to avoid SQL injection.
        List<SqlParameter> parameterList = new ArrayList<>();
        String sqlStatement = "INSERT INTO Movies
VALUES( :id , :title, :year)";

        // Create the parameters.
        SqlParameter idParam = SqlParameter.builder()
            .name("id")
            .value(String.valueOf(t))
            .build();

        SqlParameter titleParam= SqlParameter.builder()
            .name("title")
            .value(title)
            .build();

        SqlParameter yearParam = SqlParameter.builder()
            .name("year")
            .value(String.valueOf(year))
```

```
        .build();
parameterList.add(idParam);
parameterList.add(titleParam);
parameterList.add(yearParam);

try {
    ExecuteStatementRequest insertStatementRequest =
ExecuteStatementRequest.builder()
        .clusterIdentifier(clusterId)
        .sql(sqlStatement)
        .database(databaseName)
        .dbUser(userName)
        .parameters(parameterList)
        .build();

    redshiftDataClient.executeStatement(insertStatementRequest);
    System.out.println("Inserted: " + title + " (" + year + ")");
    t++;

} catch (RedshiftDataException e) {
    System.err.println("Error inserting data: " + e.getMessage());
    System.exit(1);
}
}
System.out.println(t + " records were added to the Movies table. ");
}

public static void checkStatement(RedshiftDataClient redshiftDataClient, String
sqlId) {
    try {
        DescribeStatementRequest statementRequest =
DescribeStatementRequest.builder()
            .id(sqlId)
            .build();

        String status;
        while (true) {
            DescribeStatementResponse response =
redshiftDataClient.describeStatement(statementRequest);
            status = response.statusAsString();
            System.out.println("..." + status);

            if (status.compareTo("FAILED") == 0 ) {
                System.out.println("The Query Failed. Ending program");
            }
        }
    }
}
```

```
        System.exit(1);

        } else if (status.compareTo("FINISHED") == 0) {
            break;
        }
        TimeUnit.SECONDS.sleep(1);
    }

    System.out.println("The statement is finished!");

} catch (RedshiftDataException | InterruptedException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}

}

public static void modifyCluster(RedshiftClient redshiftClient, String
clusterId) {
    try {
        ModifyClusterRequest modifyClusterRequest =
ModifyClusterRequest.builder()
            .clusterIdentifier(clusterId)
            .preferredMaintenanceWindow("wed:07:30-wed:08:00")
            .build();

        ModifyClusterResponse clusterResponse =
redshiftClient.modifyCluster(modifyClusterRequest);
        System.out.println("The modified cluster was successfully modified and
has "
            + clusterResponse.cluster().preferredMaintenanceWindow() + " as the
maintenance window");

    } catch (RedshiftException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static String queryMoviesByYear(RedshiftDataClient redshiftDataClient,
                                        String database,
                                        String dbUser,
                                        int year,
                                        String clusterId) {
```

```
try {
    String sqlStatement = " SELECT * FROM Movies WHERE year = :year";
    SqlParameter yearParam= SqlParameter.builder()
        .name("year")
        .value(String.valueOf(year))
        .build();

    ExecuteStatementRequest statementRequest =
ExecuteStatementRequest.builder()
        .clusterIdentifier(clusterId)
        .database(database)
        .dbUser(dbUser)
        .parameters(yearParam)
        .sql(sqlStatement)
        .build();

    ExecuteStatementResponse response =
redshiftDataClient.executeStatement(statementRequest);
    return response.id();

} catch (RedshiftDataException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
return "";
}

public static void getResults(RedshiftDataClient redshiftDataClient, String
statementId) {
    try {
        GetStatementResultRequest resultRequest =
GetStatementResultRequest.builder()
            .id(statementId)
            .build();

        // Extract and print the field values using streams.
        GetStatementResultResponse response =
redshiftDataClient.getStatementResult(resultRequest);
        response.records().stream()
            .flatMap(List::stream)
            .map(Field::stringValue)
            .filter(value -> value != null)
            .forEach(value -> System.out.println("The Movie title field is " +
value));
    }
}
```

```
    } catch (RedshiftDataException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void waitForClusterReady(RedshiftClient redshiftClient, String
clusterId) {
    boolean clusterReady = false;
    String clusterReadyStr;
    System.out.println("Waiting for cluster to become available. This may take a
few mins.");
    try {
        DescribeClustersRequest clustersRequest =
DescribeClustersRequest.builder()
            .clusterIdentifier(clusterId)
            .build();
        long startTime = System.currentTimeMillis();

        // Loop until the cluster is ready.
        while (!clusterReady) {
            DescribeClustersResponse clusterResponse =
redshiftClient.describeClusters(clustersRequest);
            List<Cluster> clusterList = clusterResponse.clusters();
            for (Cluster cluster : clusterList) {
                clusterReadyStr = cluster.clusterStatus();
                if (clusterReadyStr.contains("available"))
                    clusterReady = true;
                else {
                    long elapsedTimeMillis = System.currentTimeMillis() -
startTime;

                    long elapsedSeconds = elapsedTimeMillis / 1000;
                    long minutes = elapsedSeconds / 60;
                    long seconds = elapsedSeconds % 60;

                    System.out.printf("Elapsed Time: %02d:%02d - Waiting for
cluster... %n", minutes, seconds);
                    TimeUnit.SECONDS.sleep(5);
                }
            }
        }

        long elapsedTimeMillis = System.currentTimeMillis() - startTime;
```

```
        long elapsedSeconds = elapsedTimeMillis / 1000;
        long minutes = elapsedSeconds / 60;
        long seconds = elapsedSeconds % 60;

        System.out.println(String.format("Cluster is available! Total Elapsed
Time: %02d:%02d", minutes, seconds));

    } catch (RedshiftException | InterruptedException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void createTable(RedshiftDataClient redshiftDataClient, String
clusterId, String databaseName, String userName) {
    try {
        ExecuteStatementRequest createTableRequest =
ExecuteStatementRequest.builder()
            .clusterIdentifier(clusterId)
            .dbUser(userName)
            .database(databaseName)
            .sql("CREATE TABLE Movies ("
                + "id INT PRIMARY KEY, "
                + "title VARCHAR(100), "
                + "year INT)")
            .build();

        redshiftDataClient.executeStatement(createTableRequest);
        System.out.println("Table created: Movies");

    } catch (RedshiftDataException e) {
        System.err.println("Error creating table: " + e.getMessage());
        System.exit(1);
    }
}

public static void createCluster(RedshiftClient redshiftClient, String
clusterId, String masterUsername,
                                String masterUserPassword) {
    try {
        CreateClusterRequest clusterRequest = CreateClusterRequest.builder()
            .clusterIdentifier(clusterId)
            .masterUsername(masterUsername)
            .masterUserPassword(masterUserPassword)
```

```
        .nodeType("ra3.4xlarge")
        .publiclyAccessible(true)
        .numberOfNodes(2)
        .build();

        CreateClusterResponse clusterResponse =
redshiftClient.createCluster(clusterRequest);
        System.out.println("Created cluster " +
clusterResponse.cluster().clusterIdentifier());

    } catch (RedshiftException e) {

        System.err.println(e.getMessage());
        System.exit(1);
    }
}
}
```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella Documentazione di riferimento delle API AWS SDK for Java 2.x .
  - [createCluster](#)
  - [Descrivi i cluster](#)
  - [Descrivi la dichiarazione](#)
  - [Esegui dichiarazione](#)
  - [getStatementResult](#)
  - [listDatabasesPaginator](#)
  - [Modifica cluster](#)

## Esempi di Amazon Rekognition con SDK for Java 2.x

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando Amazon AWS SDK for Java 2.x Rekognition.

Le operazioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le operazioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati e negli esempi tra servizi.



Scenari: esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio.

Ogni esempio include un collegamento a GitHub, dove puoi trovare istruzioni su come configurare ed eseguire il codice nel contesto.

## Argomenti

- [Azioni](#)
- [Scenari](#)

## Azioni

### CompareFaces

Il seguente esempio di codice mostra come utilizzare CompareFaces.

Per ulteriori informazioni, consulta [Confronto dei volti nelle immagini](#).

## SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import software.amazon.awssdk.services.rekognition.model.Image;
import software.amazon.awssdk.services.rekognition.model.CompareFacesRequest;
import software.amazon.awssdk.services.rekognition.model.CompareFacesResponse;
import software.amazon.awssdk.services.rekognition.model.CompareFacesMatch;
import software.amazon.awssdk.services.rekognition.model.ComparedFace;
import software.amazon.awssdk.services.rekognition.model.BoundingBox;
import software.amazon.awssdk.core.SdkBytes;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.util.List;
```

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CompareFaces {
    public static void main(String[] args) {
        final String usage = ""

            Usage:    <pathSource> <pathTarget>

            Where:
                pathSource - The path to the source image (for example, C:\\AWS\\
\\pic1.png).\\s
                pathTarget - The path to the target image (for example, C:\\AWS\\
\\pic2.png).\\s

            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        Float similarityThreshold = 70F;
        String sourceImage = args[0];
        String targetImage = args[1];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
            .region(region)
            .build();

        compareTwoFaces(rekClient, similarityThreshold, sourceImage, targetImage);
        rekClient.close();
    }

    public static void compareTwoFaces(RekognitionClient rekClient, Float
similarityThreshold, String sourceImage,
        String targetImage) {
        try {
            InputStream sourceStream = new FileInputStream(sourceImage);
            InputStream tarStream = new FileInputStream(targetImage);
```

```
    SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);
    SdkBytes targetBytes = SdkBytes.fromInputStream(targetStream);

    // Create an Image object for the source image.
    Image souImage = Image.builder()
        .bytes(sourceBytes)
        .build();

    Image tarImage = Image.builder()
        .bytes(targetBytes)
        .build();

    CompareFacesRequest facesRequest = CompareFacesRequest.builder()
        .sourceImage(souImage)
        .targetImage(tarImage)
        .similarityThreshold(similarityThreshold)
        .build();

    // Compare the two images.
    CompareFacesResponse compareFacesResult =
rekClient.compareFaces(facesRequest);
    List<CompareFacesMatch> faceDetails = compareFacesResult.faceMatches();
    for (CompareFacesMatch match : faceDetails) {
        ComparedFace face = match.face();
        BoundingBox position = face.boundingBox();
        System.out.println("Face at " + position.left().toString()
            + " " + position.top()
            + " matches with " + face.confidence().toString()
            + "% confidence.");
    }
    List<ComparedFace> uncompered = compareFacesResult.unmatchedFaces();
    System.out.println("There was " + uncompered.size() + " face(s) that did
not match");
    System.out.println("Source image rotation: " +
compareFacesResult.sourceImageOrientationCorrection());
    System.out.println("target image rotation: " +
compareFacesResult.targetImageOrientationCorrection());

    } catch (RekognitionException | FileNotFoundException e) {
        System.out.println("Failed to load source image " + sourceImage);
        System.exit(1);
    }
}
```

```
}
```

- Per i dettagli sull'API, consulta la [CompareFaces](#) sezione AWS SDK for Java 2.x API Reference.

## CreateCollection

Il seguente esempio di codice mostra come utilizzare `CreateCollection`.

Per ulteriori informazioni, consulta [Creazione di una raccolta](#).

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.CreateCollectionResponse;
import software.amazon.awssdk.services.rekognition.model.CreateCollectionRequest;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateCollection {
    public static void main(String[] args) {
        final String usage = ""

                Usage:    <collectionName>\s

                Where:
                    collectionName - The name of the collection.\s
    }
}
```

```
        """);

    if (args.length != 1) {
        System.out.println(usage);
        System.exit(1);
    }

    String collectionId = args[0];
    Region region = Region.US_EAST_1;
    RekognitionClient rekClient = RekognitionClient.builder()
        .region(region)
        .build();

    System.out.println("Creating collection: " + collectionId);
    createMyCollection(rekClient, collectionId);
    rekClient.close();
}

public static void createMyCollection(RekognitionClient rekClient, String
collectionId) {
    try {
        CreateCollectionRequest collectionRequest =
CreateCollectionRequest.builder()
            .collectionId(collectionId)
            .build();

        CreateCollectionResponse collectionResponse =
rekClient.createCollection(collectionRequest);
        System.out.println("CollectionArn: " +
collectionResponse.collectionArn());
        System.out.println("Status code: " +
collectionResponse.statusCode().toString());

    } catch (RekognitionException e) {
        System.out.println(e.getMessage());
        System.exit(1);
    }
}
}
```

- Per i dettagli sull'API, consulta la [CreateCollection](#) sezione AWS SDK for Java 2.x API Reference.

## DeleteCollection

Il seguente esempio di codice mostra come utilizzare `DeleteCollection`.

Per ulteriori informazioni, consulta [Eliminazione di una raccolta](#).

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.DeleteCollectionRequest;
import software.amazon.awssdk.services.rekognition.model.DeleteCollectionResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteCollection {
    public static void main(String[] args) {
        final String usage = ""

                Usage:    <collectionId>\s

                Where:
                    collectionId - The id of the collection to delete.\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }
    }
}
```

```
String collectionId = args[0];
Region region = Region.US_EAST_1;
RekognitionClient rekClient = RekognitionClient.builder()
    .region(region)
    .build();

System.out.println("Deleting collection: " + collectionId);
deleteMyCollection(rekClient, collectionId);
rekClient.close();
}

public static void deleteMyCollection(RekognitionClient rekClient, String
collectionId) {
    try {
        DeleteCollectionRequest deleteCollectionRequest =
DeleteCollectionRequest.builder()
            .collectionId(collectionId)
            .build();

        DeleteCollectionResponse deleteCollectionResponse =
rekClient.deleteCollection(deleteCollectionRequest);
        System.out.println(collectionId + ": " +
deleteCollectionResponse.statusCode().toString());

    } catch (RekognitionException e) {
        System.out.println(e.getMessage());
        System.exit(1);
    }
}
}
```


- Per i dettagli sull'API, consulta la [DeleteCollection](#) sezione AWS SDK for Java 2.x API Reference.

## DeleteFaces

Il seguente esempio di codice mostra come utilizzare DeleteFaces.

Per ulteriori informazioni, consulta [Eliminazione dei volti da una raccolta](#).

## SDK per Java 2.x

 Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.DeleteFacesRequest;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteFacesFromCollection {
    public static void main(String[] args) {
        final String usage = ""

                Usage:    <collectionId> <faceId>\s

                Where:
                    collectionId - The id of the collection from which faces are
deleted.\s

                    faceId - The id of the face to delete.\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String collectionId = args[0];
        String faceId = args[1];
        Region region = Region.US_EAST_1;
```



```
    RekognitionClient rekClient = RekognitionClient.builder()
        .region(region)
        .build();

    System.out.println("Deleting collection: " + collectionId);
    deleteFacesCollection(rekClient, collectionId, faceId);
    rekClient.close();
}

public static void deleteFacesCollection(RekognitionClient rekClient,
    String collectionId,
    String faceId) {

    try {
        DeleteFacesRequest deleteFacesRequest = DeleteFacesRequest.builder()
            .collectionId(collectionId)
            .faceIds(faceId)
            .build();

        rekClient.deleteFaces(deleteFacesRequest);
        System.out.println("The face was deleted from the collection.");

    } catch (RekognitionException e) {
        System.out.println(e.getMessage());
        System.exit(1);
    }
}
}
```

- Per i dettagli sull'API, consulta la [DeleteFaces](#) sezione AWS SDK for Java 2.x API Reference.

## DescribeCollection

Il seguente esempio di codice mostra come utilizzare `DescribeCollection`.

Per ulteriori informazioni, consulta [Descrizione di una raccolta](#).

## SDK per Java 2.x

 Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.DescribeCollectionRequest;
import software.amazon.awssdk.services.rekognition.model.DescribeCollectionResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DescribeCollection {
    public static void main(String[] args) {
        final String usage = ""

            Usage:    <collectionName>

            Where:
                collectionName - The name of the Amazon Rekognition collection.\s
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String collectionName = args[0];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
            .region(region)
            .build();
```

```
        describeColl(rekClient, collectionName);
        rekClient.close();
    }

    public static void describeColl(RekognitionClient rekClient, String
collectionName) {
        try {
            DescribeCollectionRequest describeCollectionRequest =
DescribeCollectionRequest.builder()
                .collectionId(collectionName)
                .build();

            DescribeCollectionResponse describeCollectionResponse = rekClient
                .describeCollection(describeCollectionRequest);

            System.out.println("Collection Arn : " +
describeCollectionResponse.collectionARN());
            System.out.println("Created : " +
describeCollectionResponse.creationTimestamp().toString());

        } catch (RekognitionException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Per i dettagli sull'API, consulta la [DescribeCollection](#) sezione AWS SDK for Java 2.x API Reference.

## DetectFaces

Il seguente esempio di codice mostra come utilizzare DetectFaces.

Per ulteriori informazioni, consulta [Rilevamento dei volti in un'immagine](#).

## SDK per Java 2.x

 Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import software.amazon.awssdk.services.rekognition.model.DetectFacesRequest;
import software.amazon.awssdk.services.rekognition.model.DetectFacesResponse;
import software.amazon.awssdk.services.rekognition.model.Image;
import software.amazon.awssdk.services.rekognition.model.Attribute;
import software.amazon.awssdk.services.rekognition.model.FaceDetail;
import software.amazon.awssdk.services.rekognition.model.AgeRange;
import software.amazon.awssdk.core.SdkBytes;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DetectFaces {
    public static void main(String[] args) {
        final String usage = ""

            Usage:    <sourceImage>

            Where:
                sourceImage - The path to the image (for example, C:\\AWS\\
\\pic1.png).\\s
            """;
```

```
    if (args.length != 1) {
        System.out.println(usage);
        System.exit(1);
    }

    String sourceImage = args[0];
    Region region = Region.US_EAST_1;
    RekognitionClient rekClient = RekognitionClient.builder()
        .region(region)
        .build();

    detectFacesinImage(rekClient, sourceImage);
    rekClient.close();
}

public static void detectFacesinImage(RekognitionClient rekClient, String
sourceImage) {
    try {
        InputStream sourceStream = new FileInputStream(sourceImage);
        SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);

        // Create an Image object for the source image.
        Image souImage = Image.builder()
            .bytes(sourceBytes)
            .build();

        DetectFacesRequest facesRequest = DetectFacesRequest.builder()
            .attributes(Attribute.ALL)
            .image(souImage)
            .build();

        DetectFacesResponse facesResponse = rekClient.detectFaces(facesRequest);
        List<FaceDetail> faceDetails = facesResponse.faceDetails();
        for (FaceDetail face : faceDetails) {
            AgeRange ageRange = face.ageRange();
            System.out.println("The detected face is estimated to be between "
                + ageRange.low().toString() + " and " +
ageRange.high().toString()
                + " years old.");

            System.out.println("There is a smile : " +
face.smile().value().toString());
        }
    }
}
```

```
        } catch (RekognitionException | FileNotFoundException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Per i dettagli sull'API, consulta la [DetectFaces](#) sezione AWS SDK for Java 2.x API Reference.

## DetectLabels

Il seguente esempio di codice mostra come utilizzare `DetectLabels`.

Per ulteriori informazioni, consulta [Rilevamento delle etichette in un'immagine](#).

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.Image;
import software.amazon.awssdk.services.rekognition.model.DetectLabelsRequest;
import software.amazon.awssdk.services.rekognition.model.DetectLabelsResponse;
import software.amazon.awssdk.services.rekognition.model.Label;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
```

```
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class DetectLabels {
    public static void main(String[] args) {
        final String usage = ""

            Usage:    <sourceImage>

            Where:
                sourceImage - The path to the image (for example, C:\\AWS\\
\\pic1.png).\\s
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String sourceImage = args[0];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
            .region(region)
            .build();

        detectImageLabels(rekClient, sourceImage);
        rekClient.close();
    }

    public static void detectImageLabels(RekognitionClient rekClient, String
sourceImage) {
        try {
            InputStream sourceStream = new FileInputStream(sourceImage);
            SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);

            // Create an Image object for the source image.
            Image souImage = Image.builder()
                .bytes(sourceBytes)
                .build();

            DetectLabelsRequest detectLabelsRequest = DetectLabelsRequest.builder()
                .image(souImage)
                .maxLabels(10)
                .build();
```

```
        DetectLabelsResponse labelsResponse =
rekClient.detectLabels(detectLabelsRequest);
        List<Label> labels = labelsResponse.labels();
        System.out.println("Detected labels for the given photo");
        for (Label label : labels) {
            System.out.println(label.name() + ": " +
label.confidence().toString());
        }

    } catch (RekognitionException | FileNotFoundException e) {
        System.out.println(e.getMessage());
        System.exit(1);
    }
}
}
```

- Per i dettagli sull'API, consulta la [DetectLabels](#) sezione AWS SDK for Java 2.x API Reference.

## DetectModerationLabels

Il seguente esempio di codice mostra come utilizzare `DetectModerationLabels`.

Per ulteriori informazioni, consulta [Rilevamento di immagini non appropriate](#).

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import software.amazon.awssdk.services.rekognition.model.Image;
import
software.amazon.awssdk.services.rekognition.model.DetectModerationLabelsRequest;
```



```
import
    software.amazon.awssdk.services.rekognition.model.DetectModerationLabelsResponse;
import software.amazon.awssdk.services.rekognition.model.ModerationLabel;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DetectModerationLabels {

    public static void main(String[] args) {
        final String usage = ""

            Usage:    <sourceImage>

            Where:
                sourceImage - The path to the image (for example, C:\\AWS\\
\pic1.png).\s
            """;

        if (args.length < 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String sourceImage = args[0];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
            .region(region)
            .build();

        detectModLabels(rekClient, sourceImage);
        rekClient.close();
    }
}
```

```

    public static void detectModLabels(RekognitionClient rekClient, String
sourceImage) {
        try {
            InputStream sourceStream = new FileInputStream(sourceImage);
            SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);
            Image souImage = Image.builder()
                .bytes(sourceBytes)
                .build();

            DetectModerationLabelsRequest moderationLabelsRequest =
DetectModerationLabelsRequest.builder()
                .image(souImage)
                .minConfidence(60F)
                .build();

            DetectModerationLabelsResponse moderationLabelsResponse = rekClient
                .detectModerationLabels(moderationLabelsRequest);
            List<ModerationLabel> labels =
moderationLabelsResponse.moderationLabels();
            System.out.println("Detected labels for image");
            for (ModerationLabel label : labels) {
                System.out.println("Label: " + label.name()
                    + "\n Confidence: " + label.confidence().toString() + "%"
                    + "\n Parent:" + label.parentName());
            }

        } catch (RekognitionException | FileNotFoundException e) {
            e.printStackTrace();
            System.exit(1);
        }
    }
}

```

- Per i dettagli sull'API, consulta la [DetectModerationLabels](#) sezione AWS SDK for Java 2.x API Reference.

## DetectText

Il seguente esempio di codice mostra come utilizzare DetectText.

Per ulteriori informazioni, consulta [Rilevamento del testo in un'immagine](#).

## SDK per Java 2.x

 Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.DetectTextRequest;
import software.amazon.awssdk.services.rekognition.model.Image;
import software.amazon.awssdk.services.rekognition.model.DetectTextResponse;
import software.amazon.awssdk.services.rekognition.model.TextDetection;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DetectText {
    public static void main(String[] args) {
        final String usage = ""

                Usage:    <sourceImage>

                Where:
                    sourceImage - The path to the image that contains text (for
example, C:\\AWS\\pic1.png).\s
                    """;

        if (args.length != 1) {
            System.out.println(usage);
        }
    }
}
```

```
        System.exit(1);
    }

    String sourceImage = args[0];
    Region region = Region.US_EAST_1;
    RekognitionClient rekClient = RekognitionClient.builder()
        .region(region)
        .build();

    detectTextLabels(rekClient, sourceImage);
    rekClient.close();
}

public static void detectTextLabels(RekognitionClient rekClient, String
sourceImage) {
    try {
        InputStream sourceStream = new FileInputStream(sourceImage);
        SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);
        Image souImage = Image.builder()
            .bytes(sourceBytes)
            .build();

        DetectTextRequest textRequest = DetectTextRequest.builder()
            .image(souImage)
            .build();

        DetectTextResponse textResponse = rekClient.detectText(textRequest);
        List<TextDetection> textCollection = textResponse.textDetections();
        System.out.println("Detected lines and words");
        for (TextDetection text : textCollection) {
            System.out.println("Detected: " + text.detectedText());
            System.out.println("Confidence: " + text.confidence().toString());
            System.out.println("Id : " + text.id());
            System.out.println("Parent Id: " + text.parentId());
            System.out.println("Type: " + text.type());
            System.out.println();
        }
    } catch (RekognitionException | FileNotFoundException e) {
        System.out.println(e.getMessage());
        System.exit(1);
    }
}
}
```

- Per i dettagli sull'API, consulta la [DetectText](#) sezione AWS SDK for Java 2.x API Reference.

## IndexFaces

Il seguente esempio di codice mostra come utilizzare `IndexFaces`.

Per ulteriori informazioni, consulta [Indicizzazione dei volti in una raccolta](#).

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.IndexFacesResponse;
import software.amazon.awssdk.services.rekognition.model.IndexFacesRequest;
import software.amazon.awssdk.services.rekognition.model.Image;
import software.amazon.awssdk.services.rekognition.model.QualityFilter;
import software.amazon.awssdk.services.rekognition.model.Attribute;
import software.amazon.awssdk.services.rekognition.model.FaceRecord;
import software.amazon.awssdk.services.rekognition.model.UnindexedFace;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import software.amazon.awssdk.services.rekognition.model.Reason;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html

```

```
*/
public class AddFacesToCollection {
    public static void main(String[] args) {

        final String usage = ""

            Usage:      <collectionId> <sourceImage>

            Where:
                collectionName - The name of the collection.
                sourceImage - The path to the image (for example, C:\\AWS\\
\\pic1.png).\\s
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String collectionId = args[0];
        String sourceImage = args[1];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
            .region(region)
            .build();

        addToCollection(rekClient, collectionId, sourceImage);
        rekClient.close();
    }

    public static void addToCollection(RekognitionClient rekClient, String
collectionId, String sourceImage) {
        try {
            InputStream sourceStream = new FileInputStream(sourceImage);
            SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);
            Image souImage = Image.builder()
                .bytes(sourceBytes)
                .build();

            IndexFacesRequest facesRequest = IndexFacesRequest.builder()
                .collectionId(collectionId)
                .image(souImage)
                .maxFaces(1)
                .qualityFilter(QualityFilter.AUTO)

```

```

        .detectionAttributes(Attribute.DEFAULT)
        .build();

    IndexFacesResponse facesResponse = rekClient.indexFaces(facesRequest);
    System.out.println("Results for the image");
    System.out.println("\n Faces indexed:");
    List<FaceRecord> faceRecords = facesResponse.faceRecords();
    for (FaceRecord faceRecord : faceRecords) {
        System.out.println("  Face ID: " + faceRecord.face().faceId());
        System.out.println("  Location:" +
faceRecord.faceDetail().boundingBox().toString());
    }

    List<UnindexedFace> unindexedFaces = facesResponse.unindexedFaces();
    System.out.println("Faces not indexed:");
    for (UnindexedFace unindexedFace : unindexedFaces) {
        System.out.println("  Location:" +
unindexedFace.faceDetail().boundingBox().toString());
        System.out.println("  Reasons:");
        for (Reason reason : unindexedFace.reasons()) {
            System.out.println("Reason: " + reason);
        }
    }

    } catch (RekognitionException | FileNotFoundException e) {
        System.out.println(e.getMessage());
        System.exit(1);
    }
}
}
}

```

- Per i dettagli sull'API, consulta la [IndexFaces](#) sezione AWS SDK for Java 2.x API Reference.

## ListCollections

Il seguente esempio di codice mostra come utilizzare `ListCollections`.

Per ulteriori informazioni, consulta [Creazione dell'elenco delle raccolte](#).

## SDK per Java 2.x

 Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.ListCollectionsRequest;
import software.amazon.awssdk.services.rekognition.model.ListCollectionsResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListCollections {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
            .region(region)
            .build();

        System.out.println("Listing collections");
        listAllCollections(rekClient);
        rekClient.close();
    }

    public static void listAllCollections(RekognitionClient rekClient) {
        try {
            ListCollectionsRequest listCollectionsRequest =
ListCollectionsRequest.builder()
                .maxResults(10)
                .build();
```



```
        ListCollectionsResponse response =
rekClient.listCollections(listCollectionsRequest);
        List<String> collectionIds = response.collectionIds();
        for (String resultId : collectionIds) {
            System.out.println(resultId);
        }

    } catch (RekognitionException e) {
        System.out.println(e.getMessage());
        System.exit(1);
    }
}
}
```

- Per i dettagli sull'API, consulta la [ListCollections](#) sezione AWS SDK for Java 2.x API Reference.

## ListFaces

Il seguente esempio di codice mostra come utilizzare `ListFaces`.

Per ulteriori informazioni, consulta [Creazione dell'elenco dei volti in una raccolta](#).

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.Face;
import software.amazon.awssdk.services.rekognition.model.ListFacesRequest;
import software.amazon.awssdk.services.rekognition.model.ListFacesResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
```

```
*
* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class ListFacesInCollection {
    public static void main(String[] args) {
        final String usage = ""

            Usage:    <collectionId>

            Where:
                collectionId - The name of the collection.\s
            "";

        if (args.length < 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String collectionId = args[0];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
            .region(region)
            .build();

        System.out.println("Faces in collection " + collectionId);
        listFacesCollection(rekClient, collectionId);
        rekClient.close();
    }

    public static void listFacesCollection(RekognitionClient rekClient, String
collectionId) {
        try {
            ListFacesRequest facesRequest = ListFacesRequest.builder()
                .collectionId(collectionId)
                .maxResults(10)
                .build();

            ListFacesResponse facesResponse = rekClient.listFaces(facesRequest);
            List<Face> faces = facesResponse.faces();
            for (Face face : faces) {
                System.out.println("Confidence level there is a face: " +
face.confidence());
            }
        }
    }
}
```

```
        System.out.println("The face Id value is " + face.faceId());
    }

    } catch (RekognitionException e) {
        System.out.println(e.getMessage());
        System.exit(1);
    }
}
}
```

- Per i dettagli sull'API, consulta la [ListFaces](#) sezione AWS SDK for Java 2.x API Reference.

## RecognizeCelebrities

Il seguente esempio di codice mostra come utilizzare `RecognizeCelebrities`.

Per ulteriori informazioni, consulta [Riconoscimento delle celebrità in un'immagine](#).

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.core.SdkBytes;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.util.List;
import
    software.amazon.awssdk.services.rekognition.model.RecognizeCelebritiesRequest;
import
    software.amazon.awssdk.services.rekognition.model.RecognizeCelebritiesResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import software.amazon.awssdk.services.rekognition.model.Image;
import software.amazon.awssdk.services.rekognition.model.Celebrity;
```

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class RecognizeCelebrities {
    public static void main(String[] args) {
        final String usage = ""
            Usage:    <sourceImage>

            Where:
                sourceImage - The path to the image (for example, C:\\AWS\\
\\pic1.png).\\s
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String sourceImage = args[0];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
            .region(region)
            .build();

        System.out.println("Locating celebrities in " + sourceImage);
        recognizeAllCelebrities(rekClient, sourceImage);
        rekClient.close();
    }

    public static void recognizeAllCelebrities(RekognitionClient rekClient, String
sourceImage) {
        try {
            InputStream sourceStream = new FileInputStream(sourceImage);
            SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);
            Image souImage = Image.builder()
                .bytes(sourceBytes)
                .build();
        }
    }
}
```

```
        RecognizeCelebritiesRequest request =
RecognizeCelebritiesRequest.builder()
    .image(souImage)
    .build();

    RecognizeCelebritiesResponse result =
rekClient.recognizeCelebrities(request);
    List<Celebrity> celebs = result.celebrityFaces();
    System.out.println(celebs.size() + " celebrity(s) were recognized.\n");
    for (Celebrity celebrity : celebs) {
        System.out.println("Celebrity recognized: " + celebrity.name());
        System.out.println("Celebrity ID: " + celebrity.id());

        System.out.println("Further information (if available):");
        for (String url : celebrity.urls()) {
            System.out.println(url);
        }
        System.out.println();
    }
    System.out.println(result.unrecognizedFaces().size() + " face(s) were
unrecognized.");

    } catch (RekognitionException | FileNotFoundException e) {
        System.out.println(e.getMessage());
        System.exit(1);
    }
}
}
```

- Per i dettagli sull'API, consulta la [RecognizeCelebrities](#) sezione AWS SDK for Java 2.x API Reference.

## SearchFaces

Il seguente esempio di codice mostra come utilizzare `SearchFaces`.

Per ulteriori informazioni, consulta [Ricerca di un volto \(ID volto\)](#).

## SDK per Java 2.x

 Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import software.amazon.awssdk.services.rekognition.model.SearchFacesByImageRequest;
import software.amazon.awssdk.services.rekognition.model.Image;
import software.amazon.awssdk.services.rekognition.model.SearchFacesByImageResponse;
import software.amazon.awssdk.services.rekognition.model.FaceMatch;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class SearchFaceMatchingImageCollection {
    public static void main(String[] args) {
        final String usage = ""

                Usage:    <collectionId> <sourceImage>

                Where:
                    collectionId - The id of the collection. \s
                    sourceImage - The path to the image (for example, C:\\AWS\\
\\pic1.png).\s

                """;
```

```
    if (args.length != 2) {
        System.out.println(usage);
        System.exit(1);
    }

    String collectionId = args[0];
    String sourceImage = args[1];
    Region region = Region.US_EAST_1;
    RekognitionClient rekClient = RekognitionClient.builder()
        .region(region)
        .build();

    System.out.println("Searching for a face in a collections");
    searchFaceInCollection(rekClient, collectionId, sourceImage);
    rekClient.close();
}

public static void searchFaceInCollection(RekognitionClient rekClient, String
collectionId, String sourceImage) {
    try {
        InputStream sourceStream = new FileInputStream(new File(sourceImage));
        SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);
        Image souImage = Image.builder()
            .bytes(sourceBytes)
            .build();

        SearchFacesByImageRequest facesByImageRequest =
SearchFacesByImageRequest.builder()
            .image(souImage)
            .maxFaces(10)
            .faceMatchThreshold(70F)
            .collectionId(collectionId)
            .build();

        SearchFacesByImageResponse imageResponse =
rekClient.searchFacesByImage(facesByImageRequest);
        System.out.println("Faces matching in the collection");
        List<FaceMatch> faceImageMatches = imageResponse.faceMatches();
        for (FaceMatch face : faceImageMatches) {
            System.out.println("The similarity level is " + face.similarity());
            System.out.println();
        }
    }
}
```

```
        } catch (RekognitionException | FileNotFoundException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Per i dettagli sull'API, consulta la [SearchFaces](#) sezione AWS SDK for Java 2.x API Reference.

## SearchFacesByImage

Il seguente esempio di codice mostra come utilizzare `SearchFacesByImage`.

Per ulteriori informazioni, consulta [Ricerca di un volto \(immagine\)](#).

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.SearchFacesRequest;
import software.amazon.awssdk.services.rekognition.model.SearchFacesResponse;
import software.amazon.awssdk.services.rekognition.model.FaceMatch;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class SearchFaceMatchingIdCollection {
    public static void main(String[] args) {
```



```

    final String usage = ""

        Usage:    <collectionId> <sourceImage>

        Where:
            collectionId - The id of the collection. \s
            sourceImage - The path to the image (for example, C:\\AWS\\
\\pic1.png)\\.s
        """;

    if (args.length != 2) {
        System.out.println(usage);
        System.exit(1);
    }

    String collectionId = args[0];
    String faceId = args[1];
    Region region = Region.US_EAST_1;
    RekognitionClient rekClient = RekognitionClient.builder()
        .region(region)
        .build();

    System.out.println("Searching for a face in a collections");
    searchFaceById(rekClient, collectionId, faceId);
    rekClient.close();
}

public static void searchFaceById(RekognitionClient rekClient, String
collectionId, String faceId) {
    try {
        SearchFacesRequest searchFacesRequest = SearchFacesRequest.builder()
            .collectionId(collectionId)
            .faceId(faceId)
            .faceMatchThreshold(70F)
            .maxFaces(2)
            .build();

        SearchFacesResponse imageResponse =
rekClient.searchFaces(searchFacesRequest);
        System.out.println("Faces matching in the collection");
        List<FaceMatch> faceImageMatches = imageResponse.faceMatches();
        for (FaceMatch face : faceImageMatches) {
            System.out.println("The similarity level is " + face.similarity());
            System.out.println();
        }
    }
}

```

```
    }  
    } catch (RekognitionException e) {  
        System.out.println(e.getMessage());  
        System.exit(1);  
    }  
}  
}
```

- Per i dettagli sull'API, consulta la [SearchFacesByImage](#) sezione AWS SDK for Java 2.x API Reference.

## Scenari

### Rilevamento delle informazioni nei video

L'esempio di codice seguente mostra come:

- Avvia i processi di Amazon Rekognition per rilevare elementi come persone, oggetti e testo nei video.
- Controlla lo stato del processo fino al suo termine.
- Crea un output con l'elenco degli elementi rilevati da ciascun processo.

### SDK per Java 2.x

#### Note

C'è di più su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Ottieni risultati relativi alle celebrità da un video che si trova in un bucket Amazon S3.

```
import software.amazon.awssdk.regions.Region;  
import software.amazon.awssdk.services.rekognition.RekognitionClient;  
import software.amazon.awssdk.services.rekognition.model.S3Object;  
import software.amazon.awssdk.services.rekognition.model.NotificationChannel;  
import software.amazon.awssdk.services.rekognition.model.Video;
```

```

import
    software.amazon.awssdk.services.rekognition.model.StartCelebrityRecognitionResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import software.amazon.awssdk.services.rekognition.model.CelebrityRecognitionSortBy;
import software.amazon.awssdk.services.rekognition.model.VideoMetadata;
import software.amazon.awssdk.services.rekognition.model.CelebrityRecognition;
import software.amazon.awssdk.services.rekognition.model.CelebrityDetail;
import
    software.amazon.awssdk.services.rekognition.model.StartCelebrityRecognitionRequest;
import
    software.amazon.awssdk.services.rekognition.model.GetCelebrityRecognitionRequest;
import
    software.amazon.awssdk.services.rekognition.model.GetCelebrityRecognitionResponse;
import java.util.List;

/**
 * To run this code example, ensure that you perform the Prerequisites as stated
 * in the Amazon Rekognition Guide:
 * https://docs.aws.amazon.com/rekognition/latest/dg/video-analyzing-with-sqs.html
 *
 * Also, ensure that set up your development environment, including your
 * credentials.
 *
 * For information, see this documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class VideoCelebrityDetection {
    private static String startJobId = "";

    public static void main(String[] args) {
        final String usage = ""

            Usage:    <bucket> <video> <topicArn> <roleArn>

            Where:
                bucket - The name of the bucket in which the video is located
(for example, (for example, myBucket).\s
                video - The name of video (for example, people.mp4).\s
                topicArn - The ARN of the Amazon Simple Notification Service
(Amazon SNS) topic.\s
                roleArn - The ARN of the AWS Identity and Access Management (IAM)
role to use.\s

```

```
        """);

    if (args.length != 4) {
        System.out.println(usage);
        System.exit(1);
    }

    String bucket = args[0];
    String video = args[1];
    String topicArn = args[2];
    String roleArn = args[3];
    Region region = Region.US_EAST_1;
    RekognitionClient rekClient = RekognitionClient.builder()
        .region(region)
        .build();

    NotificationChannel channel = NotificationChannel.builder()
        .snsTopicArn(topicArn)
        .roleArn(roleArn)
        .build();

    startCelebrityDetection(rekClient, channel, bucket, video);
    getCelebrityDetectionResults(rekClient);
    System.out.println("This example is done!");
    rekClient.close();
}

public static void startCelebrityDetection(RekognitionClient rekClient,
    NotificationChannel channel,
    String bucket,
    String video) {
    try {
        S3Object s3Obj = S3Object.builder()
            .bucket(bucket)
            .name(video)
            .build();

        Video vidObj = Video.builder()
            .s3Object(s3Obj)
            .build();

        StartCelebrityRecognitionRequest recognitionRequest =
        StartCelebrityRecognitionRequest.builder()
            .jobTag("Celebrities")
```

```
        .notificationChannel(channel)
        .video(vid0b)
        .build();

    StartCelebrityRecognitionResponse startCelebrityRecognitionResult =
rekClient
        .startCelebrityRecognition(recognitionRequest);
    startJobId = startCelebrityRecognitionResult.jobId();

    } catch (RekognitionException e) {
        System.out.println(e.getMessage());
        System.exit(1);
    }
}

public static void getCelebrityDetectionResults(RekognitionClient rekClient) {

    try {
        String paginationToken = null;
        GetCelebrityRecognitionResponse recognitionResponse = null;
        boolean finished = false;
        String status;
        int yy = 0;

        do {
            if (recognitionResponse != null)
                paginationToken = recognitionResponse.nextToken();

            GetCelebrityRecognitionRequest recognitionRequest =
GetCelebrityRecognitionRequest.builder()
                .jobId(startJobId)
                .nextToken(paginationToken)
                .sortBy(CelebrityRecognitionSortBy.TIMESTAMP)
                .maxResults(10)
                .build();

            // Wait until the job succeeds
            while (!finished) {
                recognitionResponse =
rekClient.getCelebrityRecognition(recognitionRequest);
                status = recognitionResponse.jobStatusAsString();

                if (status.compareTo("SUCCEEDED") == 0)
                    finished = true;
            }
        }
    }
}
```

```

        else {
            System.out.println(yy + " status is: " + status);
            Thread.sleep(1000);
        }
        yy++;
    }

    finished = false;

    // Proceed when the job is done - otherwise VideoMetadata is null.
    VideoMetadata videoMetadata = recognitionResponse.videoMetadata();
    System.out.println("Format: " + videoMetadata.format());
    System.out.println("Codec: " + videoMetadata.codec());
    System.out.println("Duration: " + videoMetadata.durationMillis());
    System.out.println("FrameRate: " + videoMetadata.frameRate());
    System.out.println("Job");

    List<CelebrityRecognition> celebs =
recognitionResponse.celebrities();
    for (CelebrityRecognition celeb : celebs) {
        long seconds = celeb.timestamp() / 1000;
        System.out.print("Sec: " + seconds + " ");
        CelebrityDetail details = celeb.celebrity();
        System.out.println("Name: " + details.name());
        System.out.println("Id: " + details.id());
        System.out.println();
    }

    } while (recognitionResponse.nextToken() != null);

} catch (RekognitionException | InterruptedException e) {
    System.out.println(e.getMessage());
    System.exit(1);
}
}
}
}

```

Rileva le etichette in un video tramite un'operazione di rilevamento delle etichette.

```

import com.fasterxml.jackson.core.JsonProcessingException;
import com.fasterxml.jackson.databind.JsonMappingException;
import com.fasterxml.jackson.databind.JsonNode;

```

```

import com.fasterxml.jackson.databind.ObjectMapper;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import
    software.amazon.awssdk.services.rekognition.model.StartLabelDetectionResponse;
import software.amazon.awssdk.services.rekognition.model.NotificationChannel;
import software.amazon.awssdk.services.rekognition.model.S3Object;
import software.amazon.awssdk.services.rekognition.model.Video;
import software.amazon.awssdk.services.rekognition.model.StartLabelDetectionRequest;
import software.amazon.awssdk.services.rekognition.model.GetLabelDetectionRequest;
import software.amazon.awssdk.services.rekognition.model.GetLabelDetectionResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import software.amazon.awssdk.services.rekognition.model.LabelDetectionSortBy;
import software.amazon.awssdk.services.rekognition.model.VideoMetadata;
import software.amazon.awssdk.services.rekognition.model.LabelDetection;
import software.amazon.awssdk.services.rekognition.model.Label;
import software.amazon.awssdk.services.rekognition.model.Instance;
import software.amazon.awssdk.services.rekognition.model.Parent;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.Message;
import software.amazon.awssdk.services.sqs.model.ReceiveMessageRequest;
import software.amazon.awssdk.services.sqs.model.DeleteMessageRequest;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class VideoDetect {
    private static String startJobId = "";

    public static void main(String[] args) {
        final String usage = ""

            Usage:    <bucket> <video> <queueUrl> <topicArn> <roleArn>

            Where:
                bucket - The name of the bucket in which the video is located
                (for example, (for example, myBucket).\s
                video - The name of the video (for example, people.mp4).\s

```

```
        queueUrl- The URL of a SQS queue.\s
        topicArn - The ARN of the Amazon Simple Notification Service
(Amazon SNS) topic.\s
        roleArn - The ARN of the AWS Identity and Access Management (IAM)
role to use.\s
        """;

    if (args.length != 5) {
        System.out.println(usage);
        System.exit(1);
    }

    String bucket = args[0];
    String video = args[1];
    String queueUrl = args[2];
    String topicArn = args[3];
    String roleArn = args[4];
    Region region = Region.US_EAST_1;
    RekognitionClient rekClient = RekognitionClient.builder()
        .region(region)
        .build();

    SqsClient sqs = SqsClient.builder()
        .region(Region.US_EAST_1)
        .build();

    NotificationChannel channel = NotificationChannel.builder()
        .snsTopicArn(topicArn)
        .roleArn(roleArn)
        .build();

    startLabels(rekClient, channel, bucket, video);
    getLabelJob(rekClient, sqs, queueUrl);
    System.out.println("This example is done!");
    sqs.close();
    rekClient.close();
}

public static void startLabels(RekognitionClient rekClient,
    NotificationChannel channel,
    String bucket,
    String video) {
    try {
        S3Object s3Obj = S3Object.builder()
```



```
        .bucket(bucket)
        .name(video)
        .build();

Video vid0b = Video.builder()
    .s3Object(s3obj)
    .build();

StartLabelDetectionRequest labelDetectionRequest =
StartLabelDetectionRequest.builder()
    .jobTag("DetectingLabels")
    .notificationChannel(channel)
    .video(vid0b)
    .minConfidence(50F)
    .build();

StartLabelDetectionResponse labelDetectionResponse =
rekClient.startLabelDetection(labelDetectionRequest);
startJobId = labelDetectionResponse.jobId();

boolean ans = true;
String status = "";
int yy = 0;
while (ans) {

    GetLabelDetectionRequest detectionRequest =
GetLabelDetectionRequest.builder()
    .jobId(startJobId)
    .maxResults(10)
    .build();

    GetLabelDetectionResponse result =
rekClient.getLabelDetection(detectionRequest);
    status = result.jobStatusAsString();

    if (status.compareTo("SUCCEEDED") == 0)
        ans = false;
    else
        System.out.println(yy + " status is: " + status);

    Thread.sleep(1000);
    yy++;
}
```

```
        System.out.println(startJobId + " status is: " + status);

    } catch (RekognitionException | InterruptedException e) {
        e.getMessage();
        System.exit(1);
    }
}

public static void getLabelJob(RekognitionClient rekClient, SqsClient sqs,
String queueUrl) {
    List<Message> messages;
    ReceiveMessageRequest messageRequest = ReceiveMessageRequest.builder()
        .queueUrl(queueUrl)
        .build();

    try {
        messages = sqs.receiveMessage(messageRequest).messages();

        if (!messages.isEmpty()) {
            for (Message message : messages) {
                String notification = message.body();

                // Get the status and job id from the notification
                ObjectMapper mapper = new ObjectMapper();
                JsonNode jsonMessageTree = mapper.readTree(notification);
                JsonNode messageBodyText = jsonMessageTree.get("Message");
                ObjectMapper operationResultMapper = new ObjectMapper();
                JsonNode jsonResultTree =
operationResultMapper.readTree(messageBodyText.textValue());
                JsonNode operationJobId = jsonResultTree.get("JobId");
                JsonNode operationStatus = jsonResultTree.get("Status");
                System.out.println("Job found in JSON is " + operationJobId);

                DeleteMessageRequest deleteMessageRequest =
DeleteMessageRequest.builder()
                    .queueUrl(queueUrl)
                    .build();

                String jobId = operationJobId.textValue();
                if (startJobId.compareTo(jobId) == 0) {
                    System.out.println("Job id: " + operationJobId);
                    System.out.println("Status : " +
operationStatus.toString());
                }
            }
        }
    }
}
```

```
        if (operationStatus.asText().equals("SUCCEEDED"))
            getResultsLabels(rekClient);
        else
            System.out.println("Video analysis failed");

        sqs.deleteMessage(deleteMessageRequest);
    } else {
        System.out.println("Job received was not job " +
startJobId);
        sqs.deleteMessage(deleteMessageRequest);
    }
}

} catch (RekognitionException e) {
    e.getMessage();
    System.exit(1);
} catch (JsonMappingException e) {
    e.printStackTrace();
} catch (JsonProcessingException e) {
    e.printStackTrace();
}
}

// Gets the job results by calling GetLabelDetection
private static void getResultsLabels(RekognitionClient rekClient) {

    int maxResults = 10;
    String paginationToken = null;
    GetLabelDetectionResponse labelDetectionResult = null;

    try {
        do {
            if (labelDetectionResult != null)
                paginationToken = labelDetectionResult.nextToken();

            GetLabelDetectionRequest labelDetectionRequest =
GetLabelDetectionRequest.builder()
                .jobId(startJobId)
                .sortBy(LabelDetectionSortBy.TIMESTAMP)
                .maxResults(maxResults)
                .nextToken(paginationToken)
                .build();
```

```
        labelDetectionResult =
rekClient.getLabelDetection(labelDetectionRequest);
        VideoMetadata videoMetaData = labelDetectionResult.videoMetadata();
        System.out.println("Format: " + videoMetaData.format());
        System.out.println("Codec: " + videoMetaData.codec());
        System.out.println("Duration: " + videoMetaData.durationMillis());
        System.out.println("FrameRate: " + videoMetaData.frameRate());

        List<LabelDetection> detectedLabels = labelDetectionResult.labels();
        for (LabelDetection detectedLabel : detectedLabels) {
            long seconds = detectedLabel.timestamp();
            Label label = detectedLabel.label();
            System.out.println("Millisecond: " + seconds + " ");

            System.out.println("    Label:" + label.name());
            System.out.println("    Confidence:" +
detectedLabel.label().confidence().toString());

            List<Instance> instances = label.instances();
            System.out.println("    Instances of " + label.name());

            if (instances.isEmpty()) {
                System.out.println("        " + "None");
            } else {
                for (Instance instance : instances) {
                    System.out.println("        Confidence: " +
instance.confidence().toString());
                    System.out.println("        Bounding box: " +
instance.boundingBox().toString());
                }
            }
            System.out.println("    Parent labels for " + label.name() +
":");

            List<Parent> parents = label.parents();

            if (parents.isEmpty()) {
                System.out.println("        None");
            } else {
                for (Parent parent : parents) {
                    System.out.println("        " + parent.name());
                }
            }
            System.out.println();
        }
    }
```

```

        } while (labelDetectionResult != null &&
labelDetectionResult.nextToken() != null);

        } catch (RekognitionException e) {
            e.getMessage();
            System.exit(1);
        }
    }
}

```

### Rileva i volti in un video archiviato in un bucket Amazon S3.

```

import com.fasterxml.jackson.core.JsonProcessingException;
import com.fasterxml.jackson.databind.JsonMappingException;
import com.fasterxml.jackson.databind.JsonNode;
import com.fasterxml.jackson.databind.ObjectMapper;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import
    software.amazon.awssdk.services.rekognition.model.StartLabelDetectionResponse;
import software.amazon.awssdk.services.rekognition.model.NotificationChannel;
import software.amazon.awssdk.services.rekognition.model.S3Object;
import software.amazon.awssdk.services.rekognition.model.Video;
import software.amazon.awssdk.services.rekognition.model.StartLabelDetectionRequest;
import software.amazon.awssdk.services.rekognition.model.GetLabelDetectionRequest;
import software.amazon.awssdk.services.rekognition.model.GetLabelDetectionResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import software.amazon.awssdk.services.rekognition.model.LabelDetectionSortBy;
import software.amazon.awssdk.services.rekognition.model.VideoMetadata;
import software.amazon.awssdk.services.rekognition.model.LabelDetection;
import software.amazon.awssdk.services.rekognition.model.Label;
import software.amazon.awssdk.services.rekognition.model.Instance;
import software.amazon.awssdk.services.rekognition.model.Parent;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.Message;
import software.amazon.awssdk.services.sqs.model.ReceiveMessageRequest;
import software.amazon.awssdk.services.sqs.model.DeleteMessageRequest;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.

```

```

*
* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class VideoDetect {
    private static String startJobId = "";

    public static void main(String[] args) {
        final String usage = ""

            Usage:    <bucket> <video> <queueUrl> <topicArn> <roleArn>

            Where:
                bucket - The name of the bucket in which the video is located
(for example, (for example, myBucket).\s
                video - The name of the video (for example, people.mp4).\s
                queueUrl- The URL of a SQS queue.\s
                topicArn - The ARN of the Amazon Simple Notification Service
(Amazon SNS) topic.\s
                roleArn - The ARN of the AWS Identity and Access Management (IAM)
role to use.\s
            """;

        if (args.length != 5) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucket = args[0];
        String video = args[1];
        String queueUrl = args[2];
        String topicArn = args[3];
        String roleArn = args[4];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
            .region(region)
            .build();

        SqsClient sqs = SqsClient.builder()
            .region(Region.US_EAST_1)
            .build();

        NotificationChannel channel = NotificationChannel.builder()

```

```
        .snsTopicArn(topicArn)
        .roleArn(roleArn)
        .build();

startLabels(rekClient, channel, bucket, video);
getLabelJob(rekClient, sqs, queueUrl);
System.out.println("This example is done!");
sqs.close();
rekClient.close();
}

public static void startLabels(RecognitionClient rekClient,
    NotificationChannel channel,
    String bucket,
    String video) {
    try {
        S3Object s3obj = S3Object.builder()
            .bucket(bucket)
            .name(video)
            .build();

        Video vidObj = Video.builder()
            .s3Object(s3obj)
            .build();

        StartLabelDetectionRequest labelDetectionRequest =
StartLabelDetectionRequest.builder()
            .jobTag("DetectingLabels")
            .notificationChannel(channel)
            .video(vidObj)
            .minConfidence(50F)
            .build();

        StartLabelDetectionResponse labelDetectionResponse =
rekClient.startLabelDetection(labelDetectionRequest);
        startJobId = labelDetectionResponse.jobId();

        boolean ans = true;
        String status = "";
        int yy = 0;
        while (ans) {

            GetLabelDetectionRequest detectionRequest =
GetLabelDetectionRequest.builder()
```

```

        .jobId(startJobId)
        .maxResults(10)
        .build();

    GetLabelDetectionResponse result =
rekClient.getLabelDetection(detectionRequest);
    status = result.jobStatusAsString();

    if (status.compareTo("SUCCEEDED") == 0)
        ans = false;
    else
        System.out.println(yy + " status is: " + status);

    Thread.sleep(1000);
    yy++;
}

System.out.println(startJobId + " status is: " + status);

} catch (RekognitionException | InterruptedException e) {
    e.getMessage();
    System.exit(1);
}
}

public static void getLabelJob(RekognitionClient rekClient, SqsClient sqs,
String queueUrl) {
    List<Message> messages;
    ReceiveMessageRequest messageRequest = ReceiveMessageRequest.builder()
        .queueUrl(queueUrl)
        .build();

    try {
        messages = sqs.receiveMessage(messageRequest).messages();

        if (!messages.isEmpty()) {
            for (Message message : messages) {
                String notification = message.body();

                // Get the status and job id from the notification
                ObjectMapper mapper = new ObjectMapper();
                JsonNode jsonMessageTree = mapper.readTree(notification);
                JsonNode messageBodyText = jsonMessageTree.get("Message");
                ObjectMapper operationResultMapper = new ObjectMapper();

```



```
        JsonNode jsonResultTree =
operationResultMapper.readTree(messageBodyText.textValue());
        JsonNode operationJobId = jsonResultTree.get("JobId");
        JsonNode operationStatus = jsonResultTree.get("Status");
        System.out.println("Job found in JSON is " + operationJobId);

        DeleteMessageRequest deleteMessageRequest =
DeleteMessageRequest.builder()
            .queueUrl(queueUrl)
            .build();

        String jobId = operationJobId.textValue();
        if (startJobId.compareTo(jobId) == 0) {
            System.out.println("Job id: " + operationJobId);
            System.out.println("Status : " +
operationStatus.toString());

            if (operationStatus.asText().equals("SUCCEEDED"))
                getResultsLabels(rekClient);
            else
                System.out.println("Video analysis failed");

            sqs.deleteMessage(deleteMessageRequest);
        } else {
            System.out.println("Job received was not job " +
startJobId);

            sqs.deleteMessage(deleteMessageRequest);
        }
    }
}

} catch (RekognitionException e) {
    e.getMessage();
    System.exit(1);
} catch (JsonMappingException e) {
    e.printStackTrace();
} catch (JsonProcessingException e) {
    e.printStackTrace();
}
}

// Gets the job results by calling GetLabelDetection
private static void getResultsLabels(RekognitionClient rekClient) {
```

```
int maxResults = 10;
String paginationToken = null;
GetLabelDetectionResponse labelDetectionResult = null;

try {
    do {
        if (labelDetectionResult != null)
            paginationToken = labelDetectionResult.nextToken();

        GetLabelDetectionRequest labelDetectionRequest =
GetLabelDetectionRequest.builder()
            .jobId(startJobId)
            .sortBy(LabelDetectionSortBy.TIMESTAMP)
            .maxResults(maxResults)
            .nextToken(paginationToken)
            .build();

        labelDetectionResult =
rekClient.getLabelDetection(labelDetectionRequest);
        VideoMetadata videoMetaData = labelDetectionResult.videoMetadata();
        System.out.println("Format: " + videoMetaData.format());
        System.out.println("Codec: " + videoMetaData.codec());
        System.out.println("Duration: " + videoMetaData.durationMillis());
        System.out.println("FrameRate: " + videoMetaData.frameRate());

        List<LabelDetection> detectedLabels = labelDetectionResult.labels();
        for (LabelDetection detectedLabel : detectedLabels) {
            long seconds = detectedLabel.timestamp();
            Label label = detectedLabel.label();
            System.out.println("Millisecond: " + seconds + " ");

            System.out.println("  Label:" + label.name());
            System.out.println("  Confidence:" +
detectedLabel.label().confidence().toString());

            List<Instance> instances = label.instances();
            System.out.println("  Instances of " + label.name());

            if (instances.isEmpty()) {
                System.out.println("          " + "None");
            } else {
                for (Instance instance : instances) {
                    System.out.println("          Confidence: " +
instance.confidence().toString());
                }
            }
        }
    }
}
```

```

                System.out.println("        Bounding box: " +
instance.boundingBox().toString());
            }
        }
        System.out.println("    Parent labels for " + label.name() +
");");
        List<Parent> parents = label.parents();

        if (parents.isEmpty()) {
            System.out.println("        None");
        } else {
            for (Parent parent : parents) {
                System.out.println("        " + parent.name());
            }
        }
        System.out.println();
    }
} while (labelDetectionResult != null &&
labelDetectionResult.nextToken() != null);

    } catch (RekognitionException e) {
        e.getMessage();
        System.exit(1);
    }
}
}

```

Rileva contenuti non appropriati o offensivi in un video archiviato in un bucket Amazon S3.

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.NotificationChannel;
import software.amazon.awssdk.services.rekognition.model.S3Object;
import software.amazon.awssdk.services.rekognition.model.Video;
import
    software.amazon.awssdk.services.rekognition.model.StartContentModerationRequest;
import
    software.amazon.awssdk.services.rekognition.model.StartContentModerationResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import
    software.amazon.awssdk.services.rekognition.model.GetContentModerationResponse;

```

```
import
    software.amazon.awssdk.services.rekognition.model.GetContentModerationRequest;
import software.amazon.awssdk.services.rekognition.model.VideoMetadata;
import software.amazon.awssdk.services.rekognition.model.ContentModerationDetection;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class VideoDetectInappropriate {
    private static String startJobId = "";

    public static void main(String[] args) {

        final String usage = ""

            Usage:    <bucket> <video> <topicArn> <roleArn>

            Where:
                bucket - The name of the bucket in which the video is located
(for example, (for example, myBucket).\s
                video - The name of video (for example, people.mp4).\s
                topicArn - The ARN of the Amazon Simple Notification Service
(Amazon SNS) topic.\s
                roleArn - The ARN of the AWS Identity and Access Management (IAM)
role to use.\s
            """;

        if (args.length != 4) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucket = args[0];
        String video = args[1];
        String topicArn = args[2];
        String roleArn = args[3];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
```

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

NotificationChannel channel = NotificationChannel.builder()
        .snsTopicArn(topicArn)
        .roleArn(roleArn)
        .build();

startModerationDetection(rekClient, channel, bucket, video);
getModResults(rekClient);
System.out.println("This example is done!");
rekClient.close();
}

public static void startModerationDetection(RekognitionClient rekClient,
        NotificationChannel channel,
        String bucket,
        String video) {

    try {
        S3Object s3Obj = S3Object.builder()
                .bucket(bucket)
                .name(video)
                .build();

        Video vidObj = Video.builder()
                .s3Object(s3Obj)
                .build();

        StartContentModerationRequest modDetectionRequest =
StartContentModerationRequest.builder()
                .jobTag("Moderation")
                .notificationChannel(channel)
                .video(vidObj)
                .build();

        StartContentModerationResponse startModDetectionResult = rekClient
                .startContentModeration(modDetectionRequest);
        startJobId = startModDetectionResult.jobId();

    } catch (RekognitionException e) {
        System.out.println(e.getMessage());
        System.exit(1);
    }
}
```

```
}

public static void getModResults(RekognitionClient rekClient) {
    try {
        String paginationToken = null;
        GetContentModerationResponse modDetectionResponse = null;
        boolean finished = false;
        String status;
        int yy = 0;

        do {
            if (modDetectionResponse != null)
                paginationToken = modDetectionResponse.nextToken();

            GetContentModerationRequest modRequest =
GetContentModerationRequest.builder()
                .jobId(startJobId)
                .nextToken(paginationToken)
                .maxResults(10)
                .build();

            // Wait until the job succeeds.
            while (!finished) {
                modDetectionResponse =
rekClient.getContentModeration(modRequest);
                status = modDetectionResponse.jobStatusAsString();

                if (status.compareTo("SUCCEEDED") == 0)
                    finished = true;
                else {
                    System.out.println(yy + " status is: " + status);
                    Thread.sleep(1000);
                }
                yy++;
            }

            finished = false;

            // Proceed when the job is done - otherwise VideoMetadata is null.
            VideoMetadata videoMetaData = modDetectionResponse.videoMetadata();
            System.out.println("Format: " + videoMetaData.format());
            System.out.println("Codec: " + videoMetaData.codec());
            System.out.println("Duration: " + videoMetaData.durationMillis());
            System.out.println("FrameRate: " + videoMetaData.frameRate());
        }
    }
}
```

```

        System.out.println("Job");

        List<ContentModerationDetection> mods =
modDetectionResponse.moderationLabels();
        for (ContentModerationDetection mod : mods) {
            long seconds = mod.timestamp() / 1000;
            System.out.print("Mod label: " + seconds + " ");
            System.out.println(mod.moderationLabel().toString());
            System.out.println();
        }

        } while (modDetectionResponse != null &&
modDetectionResponse.nextToken() != null);

    } catch (RekognitionException | InterruptedException e) {
        System.out.println(e.getMessage());
        System.exit(1);
    }
}
}
}

```

Rileva segmenti di segnali d'azione tecnici e segmenti di rilevamento delle riprese in un video archiviato in un bucket Amazon S3.

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.S3Object;
import software.amazon.awssdk.services.rekognition.model.NotificationChannel;
import software.amazon.awssdk.services.rekognition.model.Video;
import software.amazon.awssdk.services.rekognition.model.StartShotDetectionFilter;
import
    software.amazon.awssdk.services.rekognition.model.StartTechnicalCueDetectionFilter;
import
    software.amazon.awssdk.services.rekognition.model.StartSegmentDetectionFilters;
import
    software.amazon.awssdk.services.rekognition.model.StartSegmentDetectionRequest;
import
    software.amazon.awssdk.services.rekognition.model.StartSegmentDetectionResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import
    software.amazon.awssdk.services.rekognition.model.GetSegmentDetectionResponse;
import software.amazon.awssdk.services.rekognition.model.GetSegmentDetectionRequest;

```

```
import software.amazon.awssdk.services.rekognition.model.VideoMetadata;
import software.amazon.awssdk.services.rekognition.model.SegmentDetection;
import software.amazon.awssdk.services.rekognition.model.TechnicalCueSegment;
import software.amazon.awssdk.services.rekognition.model.ShotSegment;
import software.amazon.awssdk.services.rekognition.model.SegmentType;
import software.amazon.awssdk.services.sqs.SqsClient;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class VideoDetectSegment {
    private static String startJobId = "";

    public static void main(String[] args) {
        final String usage = ""

            Usage:    <bucket> <video> <topicArn> <roleArn>

            Where:
                bucket - The name of the bucket in which the video is located
(for example, (for example, myBucket).\s
                video - The name of video (for example, people.mp4).\s
                topicArn - The ARN of the Amazon Simple Notification Service
(Amazon SNS) topic.\s
                roleArn - The ARN of the AWS Identity and Access Management (IAM)
role to use.\s
            """;

        if (args.length != 4) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucket = args[0];
        String video = args[1];
        String topicArn = args[2];
        String roleArn = args[3];
```



```
Region region = Region.US_EAST_1;
RekognitionClient rekClient = RekognitionClient.builder()
    .region(region)
    .build();

SqsClient sqs = SqsClient.builder()
    .region(Region.US_EAST_1)
    .build();

NotificationChannel channel = NotificationChannel.builder()
    .snsTopicArn(topicArn)
    .roleArn(roleArn)
    .build();

startSegmentDetection(rekClient, channel, bucket, video);
getSegmentResults(rekClient);
System.out.println("This example is done!");
sqs.close();
rekClient.close();
}

public static void startSegmentDetection(RekognitionClient rekClient,
    NotificationChannel channel,
    String bucket,
    String video) {
    try {
        S3Object s3obj = S3Object.builder()
            .bucket(bucket)
            .name(video)
            .build();

        Video vidObj = Video.builder()
            .s3Object(s3obj)
            .build();

        StartShotDetectionFilter cueDetectionFilter =
StartShotDetectionFilter.builder()
            .minSegmentConfidence(60F)
            .build();

        StartTechnicalCueDetectionFilter technicalCueDetectionFilter =
StartTechnicalCueDetectionFilter.builder()
            .minSegmentConfidence(60F)
            .build();
```

```
        StartSegmentDetectionFilters filters =
StartSegmentDetectionFilters.builder()
        .shotFilter(cueDetectionFilter)
        .technicalCueFilter(technicalCueDetectionFilter)
        .build();

        StartSegmentDetectionRequest segDetectionRequest =
StartSegmentDetectionRequest.builder()
        .jobTag("DetectingLabels")
        .notificationChannel(channel)
        .segmentTypes(SegmentType.TECHNICAL_CUE, SegmentType.SHOT)
        .video(vidOb)
        .filters(filters)
        .build();

        StartSegmentDetectionResponse segDetectionResponse =
rekClient.startSegmentDetection(segDetectionRequest);
        startJobId = segDetectionResponse.jobId();

    } catch (RekognitionException e) {
        e.getMessage();
        System.exit(1);
    }
}

public static void getSegmentResults(RekognitionClient rekClient) {
    try {
        String paginationToken = null;
        GetSegmentDetectionResponse segDetectionResponse = null;
        boolean finished = false;
        String status;
        int yy = 0;

        do {
            if (segDetectionResponse != null)
                paginationToken = segDetectionResponse.nextToken();

            GetSegmentDetectionRequest recognitionRequest =
GetSegmentDetectionRequest.builder()
                .jobId(startJobId)
                .nextToken(paginationToken)
                .maxResults(10)
                .build();
```

```
// Wait until the job succeeds.
while (!finished) {
    segDetectionResponse =
rekClient.getSegmentDetection(recognitionRequest);
    status = segDetectionResponse.jobStatusAsString();

    if (status.compareTo("SUCCEEDED") == 0)
        finished = true;
    else {
        System.out.println(yy + " status is: " + status);
        Thread.sleep(1000);
    }
    yy++;
}
finished = false;

// Proceed when the job is done - otherwise VideoMetadata is null.
List<VideoMetadata> videoMetaData =
segDetectionResponse.videoMetadata();
for (VideoMetadata metaData : videoMetaData) {
    System.out.println("Format: " + metaData.format());
    System.out.println("Codec: " + metaData.codec());
    System.out.println("Duration: " + metaData.durationMillis());
    System.out.println("FrameRate: " + metaData.frameRate());
    System.out.println("Job");
}

List<SegmentDetection> detectedSegments =
segDetectionResponse.segments();
for (SegmentDetection detectedSegment : detectedSegments) {
    String type = detectedSegment.type().toString();
    if (type.contains(SegmentType.TECHNICAL_CUE.toString())) {
        System.out.println("Technical Cue");
        TechnicalCueSegment segmentCue =
detectedSegment.technicalCueSegment();
        System.out.println("\tType: " + segmentCue.type());
        System.out.println("\tConfidence: " +
segmentCue.confidence().toString());
    }

    if (type.contains(SegmentType.SHOT.toString())) {
        System.out.println("Shot");
        ShotSegment segmentShot = detectedSegment.shotSegment();
    }
}
```

```

                System.out.println("\tIndex " + segmentShot.index());
                System.out.println("\tConfidence: " +
segmentShot.confidence().toString());
            }

            long seconds = detectedSegment.durationMillis();
            System.out.println("\tDuration : " + seconds + " milliseconds");
            System.out.println("\tStart time code: " +
detectedSegment.startTimecodeSMPTE());
            System.out.println("\tEnd time code: " +
detectedSegment.endTimecodeSMPTE());
            System.out.println("\tDuration time code: " +
detectedSegment.durationSMPTE());
            System.out.println();
        }

        } while (segDetectionResponse != null &&
segDetectionResponse.nextToken() != null);

    } catch (RekognitionException | InterruptedException e) {
        System.out.println(e.getMessage());
        System.exit(1);
    }
}
}

```

Rileva il testo in un video archiviato in un bucket Amazon S3.

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.S3Object;
import software.amazon.awssdk.services.rekognition.model.NotificationChannel;
import software.amazon.awssdk.services.rekognition.model.Video;
import software.amazon.awssdk.services.rekognition.model.StartTextDetectionRequest;
import software.amazon.awssdk.services.rekognition.model.StartTextDetectionResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import software.amazon.awssdk.services.rekognition.model.GetTextDetectionResponse;
import software.amazon.awssdk.services.rekognition.model.GetTextDetectionRequest;
import software.amazon.awssdk.services.rekognition.model.VideoMetadata;
import software.amazon.awssdk.services.rekognition.model.TextDetectionResult;
import java.util.List;

```

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class VideoDetectText {
    private static String startJobId = "";

    public static void main(String[] args) {
        final String usage = ""

            Usage:    <bucket> <video> <topicArn> <roleArn>

            Where:
                bucket - The name of the bucket in which the video is located
(for example, (for example, myBucket).\s
                video - The name of video (for example, people.mp4).\s
                topicArn - The ARN of the Amazon Simple Notification Service
(Amazon SNS) topic.\s
                roleArn - The ARN of the AWS Identity and Access Management (IAM)
role to use.\s
            """;

        if (args.length != 4) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucket = args[0];
        String video = args[1];
        String topicArn = args[2];
        String roleArn = args[3];

        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
            .region(region)
            .build();

        NotificationChannel channel = NotificationChannel.builder()
            .snsTopicArn(topicArn)
            .roleArn(roleArn)
```

```
        .build();

        startTextLabels(rekClient, channel, bucket, video);
        getTextResults(rekClient);
        System.out.println("This example is done!");
        rekClient.close();
    }

    public static void startTextLabels(RecognitionClient rekClient,
        NotificationChannel channel,
        String bucket,
        String video) {
        try {
            S3Object s3obj = S3Object.builder()
                .bucket(bucket)
                .name(video)
                .build();

            Video vidObj = Video.builder()
                .s3Object(s3obj)
                .build();

            StartTextDetectionRequest labelDetectionRequest =
                StartTextDetectionRequest.builder()
                    .jobTag("DetectingLabels")
                    .notificationChannel(channel)
                    .video(vidObj)
                    .build();

            StartTextDetectionResponse labelDetectionResponse =
                rekClient.startTextDetection(labelDetectionRequest);
            startJobId = labelDetectionResponse.jobId();

        } catch (RecognitionException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void getTextResults(RecognitionClient rekClient) {
        try {
            String paginationToken = null;
            GetTextDetectionResponse textDetectionResponse = null;
            boolean finished = false;
```

```
String status;
int yy = 0;

do {
    if (textDetectionResponse != null)
        paginationToken = textDetectionResponse.nextToken();

    GetTextDetectionRequest recognitionRequest =
GetTextDetectionRequest.builder()
        .jobId(startJobId)
        .nextToken(paginationToken)
        .maxResults(10)
        .build();

    // Wait until the job succeeds.
    while (!finished) {
        textDetectionResponse =
rekClient.getTextDetection(recognitionRequest);
        status = textDetectionResponse.jobStatusAsString();

        if (status.compareTo("SUCCEEDED") == 0)
            finished = true;
        else {
            System.out.println(yy + " status is: " + status);
            Thread.sleep(1000);
        }
        yy++;
    }

    finished = false;

    // Proceed when the job is done - otherwise VideoMetadata is null.
    VideoMetadata videoMetaData = textDetectionResponse.videoMetadata();
    System.out.println("Format: " + videoMetaData.format());
    System.out.println("Codec: " + videoMetaData.codec());
    System.out.println("Duration: " + videoMetaData.durationMillis());
    System.out.println("FrameRate: " + videoMetaData.frameRate());
    System.out.println("Job");

    List<TextDetectionResult> labels =
textDetectionResponse.textDetections();
    for (TextDetectionResult detectedText : labels) {
        System.out.println("Confidence: " +
detectedText.textDetection().confidence().toString());
    }
}
```

```

                System.out.println("Id : " + detectedText.textDetection().id());
                System.out.println("Parent Id: " +
detectedText.textDetection().parentId());
                System.out.println("Type: " +
detectedText.textDetection().type());
                System.out.println("Text: " +
detectedText.textDetection().detectedText());
                System.out.println();
            }

        } while (textDetectionResponse != null &&
textDetectionResponse.nextToken() != null);

        } catch (RekognitionException | InterruptedException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}

```

Rileva le persone in un video archiviato in un bucket Amazon S3.

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.S3Object;
import software.amazon.awssdk.services.rekognition.model.NotificationChannel;
import software.amazon.awssdk.services.rekognition.model.StartPersonTrackingRequest;
import software.amazon.awssdk.services.rekognition.model.Video;
import
    software.amazon.awssdk.services.rekognition.model.StartPersonTrackingResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import software.amazon.awssdk.services.rekognition.model.GetPersonTrackingResponse;
import software.amazon.awssdk.services.rekognition.model.GetPersonTrackingRequest;
import software.amazon.awssdk.services.rekognition.model.VideoMetadata;
import software.amazon.awssdk.services.rekognition.model.PersonDetection;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:

```



```
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class VideoPersonDetection {
    private static String startJobId = "";

    public static void main(String[] args) {

        final String usage = ""

            Usage:    <bucket> <video> <topicArn> <roleArn>

            Where:
                bucket - The name of the bucket in which the video is located
(for example, (for example, myBucket).\s
                video - The name of video (for example, people.mp4).\s
                topicArn - The ARN of the Amazon Simple Notification Service
(Amazon SNS) topic.\s
                roleArn - The ARN of the AWS Identity and Access Management (IAM)
role to use.\s
            """;

        if (args.length != 4) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucket = args[0];
        String video = args[1];
        String topicArn = args[2];
        String roleArn = args[3];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
            .region(region)
            .build();

        NotificationChannel channel = NotificationChannel.builder()
            .snsTopicArn(topicArn)
            .roleArn(roleArn)
            .build();

        startPersonLabels(rekClient, channel, bucket, video);
        getPersonDetectionResults(rekClient);
        System.out.println("This example is done!");
    }
}
```

```
        rekClient.close();
    }

    public static void startPersonLabels(RecognitionClient rekClient,
        NotificationChannel channel,
        String bucket,
        String video) {
        try {
            S3Object s3Obj = S3Object.builder()
                .bucket(bucket)
                .name(video)
                .build();

            Video vidObj = Video.builder()
                .s3Object(s3Obj)
                .build();

            StartPersonTrackingRequest personTrackingRequest =
                StartPersonTrackingRequest.builder()
                    .jobTag("DetectingLabels")
                    .video(vidObj)
                    .notificationChannel(channel)
                    .build();

            StartPersonTrackingResponse labelDetectionResponse =
                rekClient.startPersonTracking(personTrackingRequest);
            startJobId = labelDetectionResponse.jobId();

        } catch (RecognitionException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void getPersonDetectionResults(RecognitionClient rekClient) {
        try {
            String paginationToken = null;
            GetPersonTrackingResponse personTrackingResult = null;
            boolean finished = false;
            String status;
            int yy = 0;

            do {
                if (personTrackingResult != null)
```

```
        paginationToken = personTrackingResult.nextToken();

        GetPersonTrackingRequest recognitionRequest =
GetPersonTrackingRequest.builder()
        .jobId(startJobId)
        .nextToken(paginationToken)
        .maxResults(10)
        .build();

        // Wait until the job succeeds
        while (!finished) {

            personTrackingResult =
rekClient.getPersonTracking(recognitionRequest);
            status = personTrackingResult.jobStatusAsString();

            if (status.compareTo("SUCCEEDED") == 0)
                finished = true;
            else {
                System.out.println(yy + " status is: " + status);
                Thread.sleep(1000);
            }
            yy++;
        }

        finished = false;

        // Proceed when the job is done - otherwise VideoMetadata is null.
        VideoMetadata videoMetaData = personTrackingResult.videoMetadata();

        System.out.println("Format: " + videoMetaData.format());
        System.out.println("Codec: " + videoMetaData.codec());
        System.out.println("Duration: " + videoMetaData.durationMillis());
        System.out.println("FrameRate: " + videoMetaData.frameRate());
        System.out.println("Job");

        List<PersonDetection> detectedPersons =
personTrackingResult.persons();
        for (PersonDetection detectedPerson : detectedPersons) {
            long seconds = detectedPerson.timestamp() / 1000;
            System.out.print("Sec: " + seconds + " ");
            System.out.println("Person Identifier: " +
detectedPerson.person().index());
            System.out.println();
        }
    }
}
```

```
    }  
  
    } while (personTrackingResult != null &&  
personTrackingResult.nextToken() != null);  
  
    } catch (RekognitionException | InterruptedException e) {  
        System.out.println(e.getMessage());  
        System.exit(1);  
    }  
}  
}
```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella Documentazione di riferimento delle API AWS SDK for Java 2.x .
  - [GetCelebrityRecognition](#)
  - [GetContentModeration](#)
  - [GetLabelDetection](#)
  - [GetPersonTracking](#)
  - [GetSegmentDetection](#)
  - [GetTextDetection](#)
  - [StartCelebrityRecognition](#)
  - [StartContentModeration](#)
  - [StartLabelDetection](#)
  - [StartPersonTracking](#)
  - [StartSegmentDetection](#)
  - [StartTextDetection](#)

## Esempi di registrazione del dominio Route 53 utilizzando SDK for Java 2.x

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando la registrazione del dominio AWS SDK for Java 2.x with Route 53.

Le operazioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le operazioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati e negli esempi tra servizi.

Scenari: esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio.


Ogni esempio include un collegamento a GitHub, dove è possibile trovare istruzioni su come configurare ed eseguire il codice nel contesto.

Nozioni di base

Registrazione domini Hello Route 53

Gli esempi di codice seguenti mostrano come iniziare a usare la registrazione di domini Route 53.

SDK per Java 2.x

 Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.route53domains.Route53DomainsClient;
import software.amazon.awssdk.services.route53.model.Route53Exception;
import software.amazon.awssdk.services.route53domains.model.DomainPrice;
import software.amazon.awssdk.services.route53domains.model.ListPricesRequest;
import software.amazon.awssdk.services.route53domains.model.ListPricesResponse;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * This Java code examples performs the following operation:
 *
 * 1. Invokes ListPrices for at least one domain type, such as the "com" type
 * and displays the prices for Registration and Renewal.
 */
public class HelloRoute53 {
```

```

public static final String DASHES = new String(new char[80]).replace("\0", "-");

public static void main(String[] args) {
    final String usage = "\n" +
        "Usage:\n" +
        "    <hostedZoneId> \n\n" +
        "Where:\n" +
        "    hostedZoneId - The id value of an existing hosted zone. \n";

    if (args.length != 1) {
        System.out.println(usage);
        System.exit(1);
    }

    String domainType = args[0];
    Region region = Region.US_EAST_1;
    Route53DomainsClient route53DomainsClient = Route53DomainsClient.builder()
        .region(region)
        .build();

    System.out.println(DASHES);
    System.out.println("Invokes ListPrices for at least one domain type.");
    listPrices(route53DomainsClient, domainType);
    System.out.println(DASHES);
}

public static void listPrices(Route53DomainsClient route53DomainsClient, String
domainType) {
    try {
        ListPricesRequest pricesRequest = ListPricesRequest.builder()
            .maxItems(10)
            .tld(domainType)
            .build();

        ListPricesResponse response =
route53DomainsClient.listPrices(pricesRequest);
        List<DomainPrice> prices = response.prices();
        for (DomainPrice pr : prices) {
            System.out.println("Name: " + pr.name());
            System.out.println(
                "Registration: " + pr.registrationPrice().price() + " " +
pr.registrationPrice().currency());
            System.out.println("Renewal: " + pr.renewalPrice().price() + " " +
pr.renewalPrice().currency());
        }
    }
}

```

```
        System.out.println("Transfer: " + pr.transferPrice().price() + " " +
pr.transferPrice().currency());
        System.out.println("Transfer: " + pr.transferPrice().price() + " " +
pr.transferPrice().currency());
        System.out.println("Change Ownership: " +
pr.changeOwnershipPrice().price() + " "
        + pr.changeOwnershipPrice().currency());
        System.out.println(
        "Restoration: " + pr.restorationPrice().price() + " " +
pr.restorationPrice().currency());
        System.out.println(" ");
    }

    } catch (Route53Exception e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
}
```

- Per i dettagli sull'API, consulta la [ListPrices](#) sezione AWS SDK for Java 2.x API Reference.

## Argomenti

- [Azioni](#)
- [Scenari](#)

## Azioni

### CheckDomainAvailability

Il seguente esempio di codice mostra come utilizzare `CheckDomainAvailability`.

SDK per Java 2.x

#### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public static void checkDomainAvailability(Route53DomainsClient
route53DomainsClient, String domainSuggestion) {
    try {
        CheckDomainAvailabilityRequest availabilityRequest =
CheckDomainAvailabilityRequest.builder()
            .domainName(domainSuggestion)
            .build();

        CheckDomainAvailabilityResponse response = route53DomainsClient
            .checkDomainAvailability(availabilityRequest);
        System.out.println(domainSuggestion + " is " +
response.availability().toString());

    } catch (Route53Exception e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- Per i dettagli sull'API, consulta la [CheckDomainAvailability](#) sezione AWS SDK for Java 2.x API Reference.

## CheckDomainTransferability

Il seguente esempio di codice mostra come utilizzare `CheckDomainTransferability`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public static void checkDomainTransferability(Route53DomainsClient
route53DomainsClient, String domainSuggestion) {
    try {
        CheckDomainTransferabilityRequest transferabilityRequest =
CheckDomainTransferabilityRequest.builder()
            .domainName(domainSuggestion)
```



```
        .build();

        CheckDomainTransferabilityResponse response = route53DomainsClient
            .checkDomainTransferability(transferabilityRequest);
        System.out.println("Transferability: " +
            response.transferability().transferable().toString());

    } catch (Route53Exception e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- Per i dettagli sull'API, consulta la [CheckDomainTransferability](#) sezione AWS SDK for Java 2.x API Reference.

## GetDomainDetail

Il seguente esempio di codice mostra come utilizzare `GetDomainDetail`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public static void getDomainDetails(Route53DomainsClient route53DomainsClient,
    String domainSuggestion) {
    try {
        GetDomainDetailRequest detailRequest = GetDomainDetailRequest.builder()
            .domainName(domainSuggestion)
            .build();

        GetDomainDetailResponse response =
            route53DomainsClient.getDomainDetail(detailRequest);
        System.out.println("The contact first name is " +
            response.registrantContact().firstName());
    }
```

```
        System.out.println("The contact last name is " +
response.registrantContact().lastName());
        System.out.println("The contact org name is " +
response.registrantContact().organizationName());

    } catch (Route53Exception e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- Per i dettagli sull'API, consulta la [GetDomainDetail](#) sezione AWS SDK for Java 2.x API Reference.

## GetDomainSuggestions

Il seguente esempio di codice mostra come utilizzare `GetDomainSuggestions`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public static void listDomainSuggestions(Route53DomainsClient
route53DomainsClient, String domainSuggestion) {
    try {
        GetDomainSuggestionsRequest suggestionsRequest =
GetDomainSuggestionsRequest.builder()
            .domainName(domainSuggestion)
            .suggestionCount(5)
            .onlyAvailable(true)
            .build();

        GetDomainSuggestionsResponse response =
route53DomainsClient.getDomainSuggestions(suggestionsRequest);
        List<DomainSuggestion> suggestions = response.suggestionsList();
        for (DomainSuggestion suggestion : suggestions) {
```

```
        System.out.println("Suggestion Name: " + suggestion.domainName());
        System.out.println("Availability: " + suggestion.availability());
        System.out.println(" ");
    }

    } catch (Route53Exception e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- Per i dettagli sull'API, consulta la [GetDomainSuggestions](#) sezione AWS SDK for Java 2.x API Reference.

## GetOperationDetail

Il seguente esempio di codice mostra come utilizzare `GetOperationDetail`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public static void getOperationalDetail(Route53DomainsClient
route53DomainsClient, String operationId) {
    try {
        GetOperationDetailRequest detailRequest =
        GetOperationDetailRequest.builder()
            .operationId(operationId)
            .build();

        GetOperationDetailResponse response =
        route53DomainsClient.getOperationDetail(detailRequest);
        System.out.println("Operation detail message is " + response.message());

    } catch (Route53Exception e) {
        System.err.println(e.getMessage());
    }
}
```

```
        System.exit(1);
    }
}
```

- Per i dettagli sull'API, consulta la [GetOperationDetail](#) sezione AWS SDK for Java 2.x API Reference.

## ListDomains

Il seguente esempio di codice mostra come utilizzare `ListDomains`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).


```
public static void listDomains(Route53DomainsClient route53DomainsClient) {
    try {
        ListDomainsIterable listRes =
route53DomainsClient.listDomainsPaginator();
        listRes.stream()
            .flatMap(r -> r.domains().stream())
            .forEach(content -> System.out.println("The domain name is " +
content.domainName()));
    } catch (Route53Exception e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- Per i dettagli sull'API, consulta la [ListDomains](#) sezione AWS SDK for Java 2.x API Reference.

## ListOperations

Il seguente esempio di codice mostra come utilizzare `ListOperations`.

## SDK per Java 2.x

 Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public static void listOperations(Route53DomainsClient route53DomainsClient) {
    try {
        Date currentDate = new Date();
        LocalDateTime localDateTime =
currentDate.toInstant().atZone(ZoneId.systemDefault()).toLocalDateTime();
        ZoneOffset zoneOffset = ZoneOffset.of("+01:00");
        localDateTime = localDateTime.minusYears(1);
        Instant myTime = localDateTime.toInstant(zoneOffset);

        ListOperationsRequest operationsRequest =
ListOperationsRequest.builder()
            .submittedSince(myTime)
            .build();

        ListOperationsIterable listRes =
route53DomainsClient.listOperationsPaginator(operationsRequest);
        listRes.stream()
            .flatMap(r -> r.operations().stream())
            .forEach(content -> System.out.println(" Operation Id: " +
content.operationId() +
                " Status: " + content.statusAsString() +
                " Date: " + content.submittedDate()));

    } catch (Route53Exception e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- Per i dettagli sull'API, consulta la [ListOperations](#) sezione AWS SDK for Java 2.x API Reference.

## ListPrices

Il seguente esempio di codice mostra come utilizzare `ListPrices`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public static void listPrices(Route53DomainsClient route53DomainsClient, String
domainType) {
    try {
        ListPricesRequest pricesRequest = ListPricesRequest.builder()
            .tld(domainType)
            .build();

        ListPricesIterable listRes =
route53DomainsClient.listPricesPaginator(pricesRequest);
        listRes.stream()
            .flatMap(r -> r.prices().stream())
            .forEach(content -> System.out.println(" Name: " +
content.name() +
                " Registration: " + content.registrationPrice().price()
+ " "
                + content.registrationPrice().currency() +
                " Renewal: " + content.renewalPrice().price() + " " +
content.renewalPrice().currency()));

    } catch (Route53Exception e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- Per i dettagli sull'API, consulta la [ListPrices](#) sezione AWS SDK for Java 2.x API Reference.

## RegisterDomain

Il seguente esempio di codice mostra come utilizzare `RegisterDomain`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public static String requestDomainRegistration(Route53DomainsClient
route53DomainsClient,
    String domainSuggestion,
    String phoneNumber,
    String email,
    String firstName,
    String lastName,
    String city) {

    try {
        ContactDetail contactDetail = ContactDetail.builder()
            .contactType(ContactType.COMPANY)
            .state("LA")
            .countryCode(CountryCode.IN)
            .email(email)
            .firstName(firstName)
            .lastName(lastName)
            .city(city)
            .phoneNumber(phoneNumber)
            .organizationName("My Org")
            .addressLine1("My Address")
            .zipCode("123 123")
            .build();

        RegisterDomainRequest domainRequest = RegisterDomainRequest.builder()
            .adminContact(contactDetail)
            .registrantContact(contactDetail)
            .techContact(contactDetail)
            .domainName(domainSuggestion)
            .autoRenew(true)
            .durationInYears(1)
```

```
        .build();

        RegisterDomainResponse response =
route53DomainsClient.registerDomain(domainRequest);
        System.out.println("Registration requested. Operation Id: " +
response.operationId());
        return response.operationId();

    } catch (Route53Exception e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return "";
}
```

- Per i dettagli sull'API, consulta la [RegisterDomain](#) sezione AWS SDK for Java 2.x API Reference.

## ViewBilling

Il seguente esempio di codice mostra come utilizzare ViewBilling.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public static void listBillingRecords(Route53DomainsClient route53DomainsClient)
{
    try {
        Date currentDate = new Date();
        LocalDateTime localDateTime =
currentDate.toInstant().atZone(ZoneId.systemDefault()).toLocalDateTime();
        ZoneOffset zoneOffset = ZoneOffset.of("+01:00");
        LocalDateTime localDateTime2 = localDateTime.minusYears(1);
        Instant myStartTime = localDateTime2.toInstant(zoneOffset);
        Instant myEndTime = localDateTime.toInstant(zoneOffset);
```



```
ViewBillingRequest viewBillingRequest = ViewBillingRequest.builder()
    .start(myStartTime)
    .end(myEndTime)
    .build();

ViewBillingIterable listRes =
route53DomainsClient.viewBillingPaginator(viewBillingRequest);
listRes.stream()
    .flatMap(r -> r.billingRecords().stream())
    .forEach(content -> System.out.println(" Bill Date: " +
content.billDate() +
        " Operation: " + content.operationAsString() +
        " Price: " + content.price()));

} catch (Route53Exception e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
}
```

- Per i dettagli sull'API, consulta la [ViewBilling](#) sezione AWS SDK for Java 2.x API Reference.

## Scenari

### Nozioni di base sui domini

L'esempio di codice seguente mostra come:

- Elenca i domini correnti ed elenca le operazioni dell'anno scorso.
- Visualizza la fatturazione dell'anno scorso e visualizza i prezzi per i tipi di dominio.
- Ricevi suggerimenti sui domini.
- Verifica la disponibilità e la trasferibilità dei domini.
- Facoltativamente, richiedi la registrazione di un dominio.
- Ottieni informazioni dettagliate di un'operazione.
- Facoltativamente, ottieni informazioni dettagliate di un dominio.

## SDK per Java 2.x

 Note

C'è di più su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * This example uses pagination methods where applicable. For example, to list
 * domains, the
 * listDomainsPaginator method is used. For more information about pagination,
 * see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/pagination.html
 *
 * This Java code example performs the following operations:
 *
 * 1. List current domains.
 * 2. List operations in the past year.
 * 3. View billing for the account in the past year.
 * 4. View prices for domain types.
 * 5. Get domain suggestions.
 * 6. Check domain availability.
 * 7. Check domain transferability.
 * 8. Request a domain registration.
 * 9. Get operation details.
 * 10. Optionally, get domain details.
 */

public class Route53Scenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    public static void main(String[] args) {
        final String usage = ""
```

```

Usage:
    <domainType> <phoneNumber> <email> <domainSuggestion>
<firstName> <lastName> <city>

Where:
    domainType - The domain type (for example, com).\s
    phoneNumber - The phone number to use (for example,
+91.9966564xxx)    email - The email address to use.    domainSuggestion - The
domain suggestion (for example, findmy.accountants).\s
    firstName - The first name to use to register a domain.\s
    lastName - The last name to use to register a domain.\s
    city - the city to use to register a domain.\s
    """;

if (args.length != 7) {
    System.out.println(usage);
    System.exit(1);
}

String domainType = args[0];
String phoneNumber = args[1];
String email = args[2];
String domainSuggestion = args[3];
String firstName = args[4];
String lastName = args[5];
String city = args[6];
Region region = Region.US_EAST_1;
Route53DomainsClient route53DomainsClient = Route53DomainsClient.builder()
    .region(region)
    .build();

System.out.println(DASHES);
System.out.println("Welcome to the Amazon Route 53 domains example
scenario.");
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("1. List current domains.");
listDomains(route53DomainsClient);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("2. List operations in the past year.");

```

```
listOperations(route53DomainsClient);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("3. View billing for the account in the past year.");
listBillingRecords(route53DomainsClient);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("4. View prices for domain types.");
listPrices(route53DomainsClient, domainType);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("5. Get domain suggestions.");
listDomainSuggestions(route53DomainsClient, domainSuggestion);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("6. Check domain availability.");
checkDomainAvailability(route53DomainsClient, domainSuggestion);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("7. Check domain transferability.");
checkDomainTransferability(route53DomainsClient, domainSuggestion);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("8. Request a domain registration.");
String opId = requestDomainRegistration(route53DomainsClient,
    domainSuggestion, phoneNumber, email, firstName,
    lastName, city);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("9. Get operation details.");
getOperationalDetail(route53DomainsClient, opId);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("10. Get domain details.");
System.out.println("Note: You must have a registered domain to get
details.");
```

```
        System.out.println("Otherwise, an exception is thrown that states ");
        System.out.println("Domain xxxxxxxx not found in xxxxxxxx account.");
        getDomainDetails(route53DomainsClient, domainSuggestion);
        System.out.println(DASHES);
    }

    public static void getDomainDetails(Route53DomainsClient route53DomainsClient,
String domainSuggestion) {
        try {
            GetDomainDetailRequest detailRequest = GetDomainDetailRequest.builder()
                .domainName(domainSuggestion)
                .build();

            GetDomainDetailResponse response =
route53DomainsClient.getDomainDetail(detailRequest);
            System.out.println("The contact first name is " +
response.registrantContact().firstName());
            System.out.println("The contact last name is " +
response.registrantContact().lastName());
            System.out.println("The contact org name is " +
response.registrantContact().organizationName());

        } catch (Route53Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void getOperationalDetail(Route53DomainsClient
route53DomainsClient, String operationId) {
        try {
            GetOperationDetailRequest detailRequest =
GetOperationDetailRequest.builder()
                .operationId(operationId)
                .build();

            GetOperationDetailResponse response =
route53DomainsClient.getOperationDetail(detailRequest);
            System.out.println("Operation detail message is " + response.message());

        } catch (Route53Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

```
}

    public static String requestDomainRegistration(Route53DomainsClient
route53DomainsClient,
        String domainSuggestion,
        String phoneNumber,
        String email,
        String firstName,
        String lastName,
        String city) {

    try {
        ContactDetail contactDetail = ContactDetail.builder()
            .contactType(ContactType.COMPANY)
            .state("LA")
            .countryCode(CountryCode.IN)
            .email(email)
            .firstName(firstName)
            .lastName(lastName)
            .city(city)
            .phoneNumber(phoneNumber)
            .organizationName("My Org")
            .addressLine1("My Address")
            .zipCode("123 123")
            .build();

        RegisterDomainRequest domainRequest = RegisterDomainRequest.builder()
            .adminContact(contactDetail)
            .registrantContact(contactDetail)
            .techContact(contactDetail)
            .domainName(domainSuggestion)
            .autoRenew(true)
            .durationInYears(1)
            .build();

        RegisterDomainResponse response =
route53DomainsClient.registerDomain(domainRequest);
        System.out.println("Registration requested. Operation Id: " +
response.operationId());
        return response.operationId();

    } catch (Route53Exception e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

```
    }
    return "";
}

public static void checkDomainTransferability(Route53DomainsClient
route53DomainsClient, String domainSuggestion) {
    try {
        CheckDomainTransferabilityRequest transferabilityRequest =
CheckDomainTransferabilityRequest.builder()
            .domainName(domainSuggestion)
            .build();

        CheckDomainTransferabilityResponse response = route53DomainsClient
            .checkDomainTransferability(transferabilityRequest);
        System.out.println("Transferability: " +
response.transferability().transferable().toString());

    } catch (Route53Exception e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void checkDomainAvailability(Route53DomainsClient
route53DomainsClient, String domainSuggestion) {
    try {
        CheckDomainAvailabilityRequest availabilityRequest =
CheckDomainAvailabilityRequest.builder()
            .domainName(domainSuggestion)
            .build();

        CheckDomainAvailabilityResponse response = route53DomainsClient
            .checkDomainAvailability(availabilityRequest);
        System.out.println(domainSuggestion + " is " +
response.availability().toString());

    } catch (Route53Exception e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void listDomainSuggestions(Route53DomainsClient
route53DomainsClient, String domainSuggestion) {
```

```

        try {
            GetDomainSuggestionsRequest suggestionsRequest =
GetDomainSuggestionsRequest.builder()
                .domainName(domainSuggestion)
                .suggestionCount(5)
                .onlyAvailable(true)
                .build();

            GetDomainSuggestionsResponse response =
route53DomainsClient.getDomainSuggestions(suggestionsRequest);
            List<DomainSuggestion> suggestions = response.suggestionsList();
            for (DomainSuggestion suggestion : suggestions) {
                System.out.println("Suggestion Name: " + suggestion.domainName());
                System.out.println("Availability: " + suggestion.availability());
                System.out.println(" ");
            }

        } catch (Route53Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void listPrices(Route53DomainsClient route53DomainsClient, String
domainType) {
        try {
            ListPricesRequest pricesRequest = ListPricesRequest.builder()
                .tld(domainType)
                .build();

            ListPricesIterable listRes =
route53DomainsClient.listPricesPaginator(pricesRequest);
            listRes.stream()
                .flatMap(r -> r.prices().stream())
                .forEach(content -> System.out.println(" Name: " +
content.name() +
                    " Registration: " + content.registrationPrice().price()
+ " "
                    + content.registrationPrice().currency() +
                    " Renewal: " + content.renewalPrice().price() + " " +
content.renewalPrice().currency()));

        } catch (Route53Exception e) {
            System.err.println(e.getMessage());
        }
    }

```



```
        System.exit(1);
    }
}

public static void listBillingRecords(Route53DomainsClient route53DomainsClient)
{
    try {
        Date currentDate = new Date();
        LocalDateTime localDateTime =
currentDate.toInstant().atZone(ZoneId.systemDefault()).toLocalDateTime();
        ZoneOffset zoneOffset = ZoneOffset.of("+01:00");
        LocalDateTime localDateTime2 = localDateTime.minusYears(1);
        Instant myStartTime = localDateTime2.toInstant(zoneOffset);
        Instant myEndTime = localDateTime.toInstant(zoneOffset);

        ViewBillingRequest viewBillingRequest = ViewBillingRequest.builder()
            .start(myStartTime)
            .end(myEndTime)
            .build();

        ViewBillingIterable listRes =
route53DomainsClient.viewBillingPaginator(viewBillingRequest);
        listRes.stream()
            .flatMap(r -> r.billingRecords().stream())
            .forEach(content -> System.out.println(" Bill Date:: " +
content.billDate() +
                " Operation: " + content.operationAsString() +
                " Price: " + content.price()));

    } catch (Route53Exception e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void listOperations(Route53DomainsClient route53DomainsClient) {
    try {
        Date currentDate = new Date();
        LocalDateTime localDateTime =
currentDate.toInstant().atZone(ZoneId.systemDefault()).toLocalDateTime();
        ZoneOffset zoneOffset = ZoneOffset.of("+01:00");
        localDateTime = localDateTime.minusYears(1);
        Instant myTime = localDateTime.toInstant(zoneOffset);
```

```

        ListOperationsRequest operationsRequest =
ListOperationsRequest.builder()
        .submittedSince(myTime)
        .build();

        ListOperationsIterable listRes =
route53DomainsClient.listOperationsPaginator(operationsRequest);
        listRes.stream()
        .flatMap(r -> r.operations().stream())
        .forEach(content -> System.out.println(" Operation Id: " +
content.operationId() +
                " Status: " + content.statusAsString() +
                " Date: " + content.submittedDate()));

    } catch (Route53Exception e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void listDomains(Route53DomainsClient route53DomainsClient) {
    try {
        ListDomainsIterable listRes =
route53DomainsClient.listDomainsPaginator();
        listRes.stream()
        .flatMap(r -> r.domains().stream())
        .forEach(content -> System.out.println("The domain name is " +
content.domainName()));

    } catch (Route53Exception e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
}

```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella Documentazione di riferimento delle API AWS SDK for Java 2.x .
  - [CheckDomainAvailability](#)
  - [CheckDomainTransferability](#)
  - [GetDomainDetail](#)

- [GetDomainSuggestions](#)
- [GetOperationDetail](#)
- [ListDomains](#)
- [ListOperations](#)
- [ListPrices](#)
- [RegisterDomain](#)
- [ViewBilling](#)

## Esempi di Amazon S3 con SDK for Java 2.x

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando Amazon S3. AWS SDK for Java 2.x

Le operazioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le operazioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati e negli esempi tra servizi.

Scenari: esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio.

Ogni esempio include un collegamento a GitHub, dove puoi trovare istruzioni su come configurare ed eseguire il codice nel contesto.

Nozioni di base

Hello Amazon S3

Gli esempi di codice seguenti mostrano come iniziare a utilizzare Amazon S3.

SDK per Java 2.x

### Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;
```

```
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.Bucket;
import software.amazon.awssdk.services.s3.model.ListBucketsResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class HelloS3 {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        listBuckets(s3);
    }

    public static void listBuckets(S3Client s3) {
        try {
            ListBucketsResponse response = s3.listBuckets();
            List<Bucket> bucketList = response.buckets();
            bucketList.forEach(bucket -> {
                System.out.println("Bucket Name: " + bucket.name());
            });
        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Per i dettagli sull'API, consulta la [ListBuckets](#) sezione AWS SDK for Java 2.x API Reference.

## Argomenti

- [Azioni](#)
- [Scenari](#)
- [Esempi serverless](#)

## Azioni

### CopyObject

Il seguente esempio di codice mostra come utilizzare `CopyObject`.

SDK per Java 2.x

#### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Copia un oggetto usando un'interfaccia [S3Client](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.CopyObjectRequest;
import software.amazon.awssdk.services.s3.model.CopyObjectResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class CopyObject {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <objectKey> <fromBucket> <toBucket>
```

```
        Where:
            objectKey - The name of the object (for example, book.pdf).
            fromBucket - The S3 bucket name that contains the object (for
example, bucket1).
            toBucket - The S3 bucket to copy the object to (for example,
bucket2).

        """;

    if (args.length != 3) {
        System.out.println(usage);
        System.exit(1);
    }

    String objectKey = args[0];
    String fromBucket = args[1];
    String toBucket = args[2];
    System.out.format("Copying object %s from bucket %s to %s\n", objectKey,
fromBucket, toBucket);
    Region region = Region.US_EAST_1;
    S3Client s3 = S3Client.builder()
        .region(region)
        .build();

    copyBucketObject(s3, fromBucket, objectKey, toBucket);
    s3.close();
}

public static String copyBucketObject(S3Client s3, String fromBucket, String
objectKey, String toBucket) {
    CopyObjectRequest copyReq = CopyObjectRequest.builder()
        .sourceBucket(fromBucket)
        .sourceKey(objectKey)
        .destinationBucket(toBucket)
        .destinationKey(objectKey)
        .build();

    try {
        CopyObjectResponse copyRes = s3.copyObject(copyReq);
        return copyRes.copyObjectResult().toString();
    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

```
    }  
    return "";  
  }  
}
```

Usa un [S3 TransferManager](#) per [copiare un oggetto](#) da un bucket all'altro. Visualizza il [file completo](#) ed esegui il [test](#).

```
import org.slf4j.Logger;  
import org.slf4j.LoggerFactory;  
import software.amazon.awssdk.core.sync.RequestBody;  
import software.amazon.awssdk.services.s3.model.CopyObjectRequest;  
import software.amazon.awssdk.transfer.s3.S3TransferManager;  
import software.amazon.awssdk.transfer.s3.model.CompletedCopy;  
import software.amazon.awssdk.transfer.s3.model.Copy;  
import software.amazon.awssdk.transfer.s3.model.CopyRequest;  
  
import java.util.UUID;  
  
public String copyObject(S3TransferManager transferManager, String bucketName,  
    String key, String destinationBucket, String destinationKey) {  
    CopyObjectRequest copyObjectRequest = CopyObjectRequest.builder()  
        .sourceBucket(bucketName)  
        .sourceKey(key)  
        .destinationBucket(destinationBucket)  
        .destinationKey(destinationKey)  
        .build();  
  
    CopyRequest copyRequest = CopyRequest.builder()  
        .copyObjectRequest(copyObjectRequest)  
        .build();  
  
    Copy copy = transferManager.copy(copyRequest);  
  
    CompletedCopy completedCopy = copy.completionFuture().join();  
    return completedCopy.response().copyObjectResult().eTag();  
}
```

- Per i dettagli sull'API, consulta la sezione AWS SDK for Java 2.x API [CopyObjectReference](#).

## CreateBucket

Il seguente esempio di codice mostra come utilizzare `CreateBucket`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Creare un bucket.

```
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.CreateBucketRequest;
import software.amazon.awssdk.services.s3.model.HeadBucketRequest;
import software.amazon.awssdk.services.s3.model.HeadBucketResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.waiters.S3Waiter;
import java.net.URISyntaxException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class CreateBucket {
    public static void main(String[] args) throws URISyntaxException {
        final String usage = ""

                Usage:
                <bucketName>\s

                Where:
                bucketName - The name of the bucket to create. The bucket name
                must be unique, or an error occurs.
    }
}
```



```
        """;

    if (args.length != 1) {
        System.out.println(usage);
        System.exit(1);
    }

    String bucketName = args[0];
    System.out.format("Creating a bucket named %s\n", bucketName);
    Region region = Region.US_EAST_1;
    S3Client s3 = S3Client.builder()
        .region(region)
        .build();

    createBucket(s3, bucketName);
    s3.close();
}

public static void createBucket(S3Client s3Client, String bucketName) {
    try {
        S3Waiter s3Waiter = s3Client.waiter();
        CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
            .bucket(bucketName)
            .build();

        s3Client.createBucket(bucketRequest);
        HeadBucketRequest bucketRequestWait = HeadBucketRequest.builder()
            .bucket(bucketName)
            .build();

        // Wait until the bucket is created and print out the response.
        WaiterResponse<HeadBucketResponse> waiterResponse =
s3Waiter.waitUntilBucketExists(bucketRequestWait);
        waiterResponse.matched().response().ifPresent(System.out::println);
        System.out.println(bucketName + " is ready");

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

Crea un bucket con il blocco degli oggetti abilitato.

```
// Create a new Amazon S3 bucket with object lock options.
public void createBucketWithLockOptions(boolean enableObjectLock, String
bucketName) {
    S3Waiter s3Waiter = getClient().waiter();
    CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
        .bucket(bucketName)
        .objectLockEnabledForBucket(enableObjectLock)
        .build();

    getClient().createBucket(bucketRequest);
    HeadBucketRequest bucketRequestWait = HeadBucketRequest.builder()
        .bucket(bucketName)
        .build();

    // Wait until the bucket is created and print out the response.
    s3Waiter.waitUntilBucketExists(bucketRequestWait);
    System.out.println(bucketName + " is ready");
}
```

- Per i dettagli sull'API, consulta la sezione [CreateBucket AWS SDK for Java 2.x API Reference](#).

## DeleteBucket

Il seguente esempio di codice mostra come utilizzare `DeleteBucket`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
DeleteBucketRequest deleteBucketRequest = DeleteBucketRequest.builder()
    .bucket(bucket)
    .build();

s3.deleteBucket(deleteBucketRequest);
```

```
s3.close();
```

- Per i dettagli sull'API, consulta la [DeleteBucket](#) sezione AWS SDK for Java 2.x API Reference.

## DeleteBucketPolicy

Il seguente esempio di codice mostra come utilizzare DeleteBucketPolicy.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.DeleteBucketPolicyRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class DeleteBucketPolicy {
    public static void main(String[] args) {

        final String usage = ""

            Usage:
                <bucketName>

            Where:
                bucketName - The Amazon S3 bucket to delete the policy from (for
example, bucket1)."";
    }
}
```

```
    if (args.length != 1) {
        System.out.println(usage);
        System.exit(1);
    }

    String bucketName = args[0];
    System.out.format("Deleting policy from bucket: \"%s\"\n\n", bucketName);
    Region region = Region.US_EAST_1;
    S3Client s3 = S3Client.builder()
        .region(region)
        .build();

    deleteS3BucketPolicy(s3, bucketName);
    s3.close();
}

// Delete the bucket policy.
public static void deleteS3BucketPolicy(S3Client s3, String bucketName) {
    DeleteBucketPolicyRequest delReq = DeleteBucketPolicyRequest.builder()
        .bucket(bucketName)
        .build();

    try {
        s3.deleteBucketPolicy(delReq);
        System.out.println("Done!");
    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Per i dettagli sull'API, consulta la [DeleteBucketPolicy](#) sezione AWS SDK for Java 2.x API Reference.

## DeleteBucketWebsite

Il seguente esempio di codice mostra come utilizzare `DeleteBucketWebsite`.

## SDK per Java 2.x

 Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.DeleteBucketWebsiteRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class DeleteWebsiteConfiguration {
    public static void main(String[] args) {
        final String usage = ""

                Usage:      <bucketName>

                Where:
                    bucketName - The Amazon S3 bucket to delete the website
configuration from.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        System.out.format("Deleting website configuration for Amazon S3 bucket: %s
\n", bucketName);
        Region region = Region.US_EAST_1;
```

```
S3Client s3 = S3Client.builder()
    .region(region)
    .build();

deleteBucketWebsiteConfig(s3, bucketName);
System.out.println("Done!");
s3.close();
}

public static void deleteBucketWebsiteConfig(S3Client s3, String bucketName) {
    DeleteBucketWebsiteRequest delReq = DeleteBucketWebsiteRequest.builder()
        .bucket(bucketName)
        .build();

    try {
        s3.deleteBucketWebsite(delReq);

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.out.println("Failed to delete website configuration!");
        System.exit(1);
    }
}
}
```

- Per i dettagli sull'API, consulta la [DeleteBucketWebsite](#) sezione AWS SDK for Java 2.x API Reference.

## DeleteObjects

Il seguente esempio di codice mostra come utilizzare `DeleteObjects`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.model.ObjectIdentifier;
import software.amazon.awssdk.services.s3.model.Delete;
import software.amazon.awssdk.services.s3.model.DeleteObjectsRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import java.util.ArrayList;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class DeleteMultiObjects {
    public static void main(String[] args) {
        final String usage = ""

                Usage:    <bucketName>

                Where:
                    bucketName - the Amazon S3 bucket name.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        deleteBucketObjects(s3, bucketName);
        s3.close();
    }
}
```

```
public static void deleteBucketObjects(S3Client s3, String bucketName) {
    // Upload three sample objects to the specified Amazon S3 bucket.
    ArrayList<ObjectIdentifier> keys = new ArrayList<>();
    PutObjectRequest putObj;
    ObjectIdentifier objectId;

    for (int i = 0; i < 3; i++) {
        String keyName = "delete object example " + i;
        objectId = ObjectIdentifier.builder()
            .key(keyName)
            .build();

        putObj = PutObjectRequest.builder()
            .bucket(bucketName)
            .key(keyName)
            .build();

        s3.putObject(putObj, RequestBody.fromString(keyName));
        keys.add(objectId);
    }

    System.out.println(keys.size() + " objects successfully created.");

    // Delete multiple objects in one request.
    Delete del = Delete.builder()
        .objects(keys)
        .build();

    try {
        DeleteObjectsRequest multiObjectDeleteRequest =
        DeleteObjectsRequest.builder()
            .bucket(bucketName)
            .delete(del)
            .build();

        s3.deleteObjects(multiObjectDeleteRequest);
        System.out.println("Multiple objects are deleted!");

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```



```
}
```

- Per i dettagli sull'API, consulta la [DeleteObjects](#) sezione AWS SDK for Java 2.x API Reference.

## GetBucketAcl

Il seguente esempio di codice mostra come utilizzare `GetBucketAcl`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetObjectAclRequest;
import software.amazon.awssdk.services.s3.model.GetObjectAclResponse;
import software.amazon.awssdk.services.s3.model.Grant;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class GetAcl {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <bucketName> <objectKey>

                Where:
```

```

        bucketName - The Amazon S3 bucket to get the access control list
(ACL) for.
        objectKey - The object to get the ACL for.\s
        """;

    if (args.length != 2) {
        System.out.println(usage);
        System.exit(1);
    }

    String bucketName = args[0];
    String objectKey = args[1];
    System.out.println("Retrieving ACL for object: " + objectKey);
    System.out.println("in bucket: " + bucketName);
    Region region = Region.US_EAST_1;
    S3Client s3 = S3Client.builder()
        .region(region)
        .build();

    getBucketACL(s3, objectKey, bucketName);
    s3.close();
    System.out.println("Done!");
}

public static String getBucketACL(S3Client s3, String objectKey, String
bucketName) {
    try {
        GetObjectAclRequest aclReq = GetObjectAclRequest.builder()
            .bucket(bucketName)
            .key(objectKey)
            .build();

        GetObjectAclResponse aclRes = s3.getObjectAcl(aclReq);
        List<Grant> grants = aclRes.grants();
        String grantee = "";
        for (Grant grant : grants) {
            System.out.format("  %s: %s\n", grant.grantee().id(),
grant.permission());
            grantee = grant.grantee().id();
        }

        return grantee;
    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }
}

```

```
        System.exit(1);
    }

    return "";
}
}
```

- Per i dettagli sull'API, consulta la [GetBucketAc](#) sezione AWS SDK for Java 2.x API Reference.

## GetBucketPolicy

Il seguente esempio di codice mostra come utilizzare `GetBucketPolicy`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetBucketPolicyRequest;
import software.amazon.awssdk.services.s3.model.GetBucketPolicyResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class GetBucketPolicy {
    public static void main(String[] args) {
        final String usage = ""

        Usage:
```

```
        <bucketName>

        Where:
            bucketName - The Amazon S3 bucket to get the policy from.
            """";

    if (args.length != 1) {
        System.out.println(usage);
        System.exit(1);
    }

    String bucketName = args[0];
    System.out.format("Getting policy for bucket: \"%s\"\n\n", bucketName);
    Region region = Region.US_EAST_1;
    S3Client s3 = S3Client.builder()
        .region(region)
        .build();

    String polText = getPolicy(s3, bucketName);
    System.out.println("Policy Text: " + polText);
    s3.close();
}

public static String getPolicy(S3Client s3, String bucketName) {
    String policyText;
    System.out.format("Getting policy for bucket: \"%s\"\n\n", bucketName);
    GetBucketPolicyRequest policyReq = GetBucketPolicyRequest.builder()
        .bucket(bucketName)
        .build();

    try {
        GetBucketPolicyResponse policyRes = s3.getBucketPolicy(policyReq);
        policyText = policyRes.policy();
        return policyText;
    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }

    return "";
}
}
```

- Per i dettagli sull'API, consulta la [GetBucketPolicy](#) sezione AWS SDK for Java 2.x API Reference.

## GetObject

Il seguente esempio di codice mostra come utilizzare `GetObject`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Leggi i dati come array di byte utilizzando un'interfaccia [S3Client](#).

```
import software.amazon.awssdk.core.ResponseBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.GetObjectResponse;
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.OutputStream;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class GetObjectData {
    public static void main(String[] args) {
        final String usage = ""
```

```

Usage:
    <bucketName> <keyName> <path>

Where:
    bucketName - The Amazon S3 bucket name.\s
    keyName - The key name.\s
    path - The path where the file is written to.\s
    """";

if (args.length != 3) {
    System.out.println(usage);
    System.exit(1);
}

String bucketName = args[0];
String keyName = args[1];
String path = args[2];
Region region = Region.US_EAST_1;
S3Client s3 = S3Client.builder()
    .region(region)
    .build();

getObjectBytes(s3, bucketName, keyName, path);
}

public static void getObjectBytes(S3Client s3, String bucketName, String
keyName, String path) {
    try {
        GetObjectRequest objectRequest = GetObjectRequest
            .builder()
            .key(keyName)
            .bucket(bucketName)
            .build();

        ResponseBytes<GetObjectResponse> objectBytes =
s3.getObjectAsBytes(objectRequest);
        byte[] data = objectBytes.asByteArray();

        // Write the data to a local file.
        File myFile = new File(path);
        OutputStream os = new FileOutputStream(myFile);
        os.write(data);
        System.out.println("Successfully obtained bytes from an S3 object");
    }
}

```

```
        os.close();

    } catch (IOException ex) {
        ex.printStackTrace();
    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

Usa un [S3 TransferManager](#) per [scaricare un oggetto](#) in un bucket S3 in un file locale. Visualizza il [file completo](#) ed esegui il [test](#).

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.CompletedFileDownload;
import software.amazon.awssdk.transfer.s3.model.DownloadFileRequest;
import software.amazon.awssdk.transfer.s3.model.FileDownload;
import software.amazon.awssdk.transfer.s3.progress.LoggingTransferListener;

import java.io.IOException;
import java.net.URISyntaxException;
import java.net.URL;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.UUID;

public Long downloadFile(S3TransferManager transferManager, String bucketName,
                        String key, String downloadedFileWithPath) {
    DownloadFileRequest downloadFileRequest = DownloadFileRequest.builder()
        .getObjectRequest(b -> b.bucket(bucketName).key(key))
        .destination(Paths.get(downloadedFileWithPath))
        .build();

    FileDownload downloadFile =
transferManager.downloadFile(downloadFileRequest);
```

```

        CompletedFileDownload downloadResult =
downloadFile.completionFuture().join();
        logger.info("Content length [{}]",
downloadResult.response().contentType());
        return downloadResult.response().contentType();
    }

```

Leggi i tag appartenenti a un oggetto utilizzando un'interfaccia [S3Client](#).

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetObjectTaggingRequest;
import software.amazon.awssdk.services.s3.model.GetObjectTaggingResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.Tag;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class GetObjectTags {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <bucketName> <keyName>\s

                Where:
                bucketName - The Amazon S3 bucket name.\s
                keyName - A key name that represents the object.\s

                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }
    }

```



```
String bucketName = args[0];
String keyName = args[1];
Region region = Region.US_EAST_1;
S3Client s3 = S3Client.builder()
    .region(region)
    .build();

listTags(s3, bucketName, keyName);
s3.close();
}

public static void listTags(S3Client s3, String bucketName, String keyName) {
    try {
        GetObjectTaggingRequest getTaggingRequest = GetObjectTaggingRequest
            .builder()
            .key(keyName)
            .bucket(bucketName)
            .build();

        GetObjectTaggingResponse tags = s3.getObjectTagging(getTaggingRequest);
        List<Tag> tagSet = tags.tagSet();
        for (Tag tag : tagSet) {
            System.out.println(tag.key());
            System.out.println(tag.value());
        }

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

Recupera un URL per un oggetto utilizzando un'interfaccia [S3Client](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetUrlRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import java.net.URL;
```

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class GetObjectUrl {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <bucketName> <keyName>\s

            Where:
                bucketName - The Amazon S3 bucket name.
                keyName - A key name that represents the object.\s
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String keyName = args[1];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        getURL(s3, bucketName, keyName);
        s3.close();
    }

    public static void getURL(S3Client s3, String bucketName, String keyName) {
        try {
            GetUrlRequest request = GetUrlRequest.builder()
                .bucket(bucketName)
                .key(keyName)
                .build();
        }
    }
}
```

```

        URL url = s3.utilities().getUrl(request);
        System.out.println("The URL for " + keyName + " is " + url);

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}

```

Recupera un oggetto utilizzando l'oggetto client S3Presigner mediante un'interfaccia [S3Client](#).

```

import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.net.HttpURLConnection;
import java.time.Duration;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.presigner.model.GetObjectPresignRequest;
import software.amazon.awssdk.services.s3.presigner.model.PresignedGetObjectRequest;
import software.amazon.awssdk.services.s3.presigner.S3Presigner;
import software.amazon.awssdk.utils.IoUtils;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class GetObjectPresignedUrl {
    public static void main(String[] args) {
        final String USAGE = ""

                Usage:
                <bucketName> <keyName>\s

                Where:
                bucketName - The Amazon S3 bucket name.\s
    }
}

```

```

        keyName - A key name that represents a text file.\s
        """;

    if (args.length != 2) {
        System.out.println(USAGE);
        System.exit(1);
    }

    String bucketName = args[0];
    String keyName = args[1];
    Region region = Region.US_EAST_1;
    S3Presigner presigner = S3Presigner.builder()
        .region(region)
        .build();

    getPresignedUrl(presigner, bucketName, keyName);
    presigner.close();
}

public static void getPresignedUrl(S3Presigner presigner, String bucketName,
String keyName) {
    try {
        GetObjectRequest getObjectRequest = GetObjectRequest.builder()
            .bucket(bucketName)
            .key(keyName)
            .build();

        GetObjectPresignRequest getObjectPresignRequest =
GetObjectPresignRequest.builder()
            .signatureDuration(Duration.ofMinutes(60))
            .getObjectRequest(getObjectRequest)
            .build();

        PresignedGetObjectRequest presignedGetObjectRequest =
presigner.presignGetObject(getObjectPresignRequest);
        String theUrl = presignedGetObjectRequest.url().toString();
        System.out.println("Presigned URL: " + theUrl);
        HttpURLConnection connection = (HttpURLConnection)
presignedGetObjectRequest.url().openConnection();
        presignedGetObjectRequest.httpRequest().headers().forEach((header,
values) -> {
            values.forEach(value -> {
                connection.addRequestProperty(header, value);
            });
        });
    }
}

```

```

    });

    // Send any request payload that the service needs (not needed when
    // isBrowserExecutable is true).
    if (presignedGetObjectRequest.signedPayload().isPresent()) {
        connection.setDoOutput(true);

        try (InputStream signedPayload =
presignedGetObjectRequest.signedPayload().get().asInputStream();
            OutputStream httpOutputStream =
connection.getOutputStream()) {
            IoUtils.copy(signedPayload, httpOutputStream);
        }
    }

    // Download the result of executing the request.
    try (InputStream content = connection.getInputStream()) {
        System.out.println("Service returned response: ");
        IoUtils.copy(content, System.out);
    }

} catch (S3Exception | IOException e) {
    e.printStackTrace();
}
}
}
}

```

### [Ottieni un oggetto usando un ResponseTransformer oggetto e S3Client.](#)

```

import software.amazon.awssdk.core.ResponseBytes;
import software.amazon.awssdk.core.sync.ResponseTransformer;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.GetObjectResponse;
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.OutputStream;

/**

```

```
* Before running this Java V2 code example, set up your development
* environment, including your credentials.
*
* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
```

```
public class GetDataResponseTransformer {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <bucketName> <keyName> <path>

            Where:
                bucketName - The Amazon S3 bucket name.\s
                keyName - The key name.\s
                path - The path where the file is written to.\s
            """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String keyName = args[1];
        String path = args[2];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        getObjectBytes(s3, bucketName, keyName, path);
        s3.close();
    }

    public static void getObjectBytes(S3Client s3, String bucketName, String
    keyName, String path) {
        try {
            GetObjectRequest objectRequest = GetObjectRequest
                .builder()
                .key(keyName)
```

```
        .bucket(bucketName)
        .build();

    ResponseBytes<GetObjectResponse> objectBytes =
s3.getObject(objectRequest, ResponseTransformer.toBytes());
    byte[] data = objectBytes.asByteArray();

    // Write the data to a local file.
    File myFile = new File(path);
    OutputStream os = new FileOutputStream(myFile);
    os.write(data);
    System.out.println("Successfully obtained bytes from an S3 object");
    os.close();

    } catch (IOException ex) {
        ex.printStackTrace();
    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Per i dettagli sull'API, vedi [GetObject](#) in AWS SDK for Java 2.x API Reference.

## GetObjectLegalHold

Il seguente esempio di codice mostra come utilizzare `GetObjectLegalHold`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
// Get the legal hold details for an S3 object.
public ObjectLockLegalHold getObjectLegalHold(String bucketName, String
objectKey) {
    try {
```

```
GetObjectLegalHoldRequest legalHoldRequest =
GetObjectLegalHoldRequest.builder()
    .bucket(bucketName)
    .key(objectKey)
    .build();

GetObjectLegalHoldResponse response =
getClient().getObjectLegalHold(legalHoldRequest);
System.out.println("Object legal hold for " + objectKey + " in " +
bucketName +
    ":\n\tStatus: " + response.legalHold().status());
return response.legalHold();

} catch (S3Exception ex) {
    System.out.println("\tUnable to fetch legal hold: '" + ex.getMessage() +
    "'");
}

return null;
}
```

- Per i dettagli sull'API, consulta la [GetObjectLegalHold](#) sezione AWS SDK for Java 2.x API Reference.

## GetObjectLockConfiguration

Il seguente esempio di codice mostra come utilizzare `GetObjectLockConfiguration`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
// Get the object lock configuration details for an S3 bucket.
public void getBucketObjectLockConfiguration(String bucketName) {
    GetObjectLockConfigurationRequest objectLockConfigurationRequest =
GetObjectLockConfigurationRequest.builder()
```



```
        .bucket(bucketName)
        .build();

    GetObjectLockConfigurationResponse response =
getClient().getObjectLockConfiguration(objectLockConfigurationRequest);
    System.out.println("Bucket object lock config for "+bucketName+": ");
    System.out.println("\tEnabled:
"+response.getObjectLockConfiguration().getObjectLockEnabled());
    System.out.println("\tRule: "+
response.getObjectLockConfiguration().rule().defaultRetention());
    }
```

- Per i dettagli sull'API, consulta la [GetObjectLockConfiguration](#) sezione AWS SDK for Java 2.x API Reference.

## GetObjectRetention

Il seguente esempio di codice mostra come utilizzare `GetObjectRetention`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
// Get the retention period for an S3 object.
public ObjectLockRetention getObjectRetention(String bucketName, String key){
    try {
        GetObjectRetentionRequest retentionRequest =
GetObjectRetentionRequest.builder()
            .bucket(bucketName)
            .key(key)
            .build();

        GetObjectRetentionResponse response =
getClient().getObjectRetention(retentionRequest);
    }
```

```
        System.out.println("Object retention for "+key+" in "+ bucketName +":  
" + response.retention().mode() +" until "+ response.retention().retainUntilDate()  
+ ".");  
        return response.retention();  
  
    } catch (S3Exception e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        return null;  
    }  
}
```

- Per i dettagli sull'API, consulta la [GetObjectRetention](#) sezione AWS SDK for Java 2.x API Reference.

## HeadObject

Il seguente esempio di codice mostra come utilizzare `HeadObject`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Determinazione del tipo di contenuto di un oggetto.

```
import software.amazon.awssdk.regions.Region;  
import software.amazon.awssdk.services.s3.S3Client;  
import software.amazon.awssdk.services.s3.model.HeadObjectRequest;  
import software.amazon.awssdk.services.s3.model.HeadObjectResponse;  
import software.amazon.awssdk.services.s3.model.S3Exception;  
  
/**  
 * Before running this Java V2 code example, set up your development  
 * environment, including your credentials.  
 *  
 * For more information, see the following documentation topic:  
 *  
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html */
```

```
*/
public class GetObjectContentType {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <bucketName> <keyName>>

            Where:
                bucketName - The Amazon S3 bucket name.\s
                keyName - The key name.\s
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String keyName = args[1];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        getContentType(s3, bucketName, keyName);
        s3.close();
    }

    public static void getContentType(S3Client s3, String bucketName, String
keyName) {
        try {
            HeadObjectRequest objectRequest = HeadObjectRequest.builder()
                .key(keyName)
                .bucket(bucketName)
                .build();

            HeadObjectResponse objectHead = s3.headObject(objectRequest);
            String type = objectHead.contentType();
            System.out.println("The object content type is " + type);

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

```

    }
  }
}

```

## Controllo dello stato di ripristino di un oggetto.

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.HeadObjectRequest;
import software.amazon.awssdk.services.s3.model.HeadObjectResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;

public class GetObjectRestoreStatus {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <bucketName> <keyName>\s

                Where:
                bucketName - The Amazon S3 bucket name.\s
                keyName - A key name that represents the object.\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String keyName = args[1];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        checkStatus(s3, bucketName, keyName);
        s3.close();
    }

    public static void checkStatus(S3Client s3, String bucketName, String keyName) {
        try {

```

```

        HeadObjectRequest headObjectRequest = HeadObjectRequest.builder()
            .bucket(bucketName)
            .key(keyName)
            .build();

        HeadObjectResponse response = s3.headObject(headObjectRequest);
        System.out.println("The Amazon S3 object restoration status is " +
response.restore());

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}

```

- Per i dettagli sull'API, consulta la [HeadObject](#) sezione AWS SDK for Java 2.x API Reference.

## ListBuckets

Il seguente esempio di codice mostra come utilizzare `ListBuckets`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.Bucket;
import software.amazon.awssdk.services.s3.model.ListBucketsResponse;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:

```

```
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class ListBuckets {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        listAllBuckets(s3);
    }
    public static void listAllBuckets(S3Client s3) {
        ListBucketsResponse response = s3.listBuckets();
        List<Bucket> bucketList = response.buckets();
        for (Bucket bucket: bucketList) {
            System.out.println("Bucket name "+bucket.name());
        }
    }
}
```

- Per i dettagli sull'API, consulta la [ListBuckets](#) sezione AWS SDK for Java 2.x API Reference.

## ListMultipartUploads

Il seguente esempio di codice mostra come utilizzare `ListMultipartUploads`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.ListMultipartUploadsRequest;
import software.amazon.awssdk.services.s3.model.ListMultipartUploadsResponse;
import software.amazon.awssdk.services.s3.model.MultipartUpload;
```

```
import software.amazon.awssdk.services.s3.model.S3Exception;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class ListMultipartUploads {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <bucketName>\s

                Where:
                bucketName - The name of the Amazon S3 bucket where an in-
                progress multipart upload is occurring.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();
        listUploads(s3, bucketName);
        s3.close();
    }

    public static void listUploads(S3Client s3, String bucketName) {
        try {
            ListMultipartUploadsRequest listMultipartUploadsRequest =
            ListMultipartUploadsRequest.builder()
                .bucket(bucketName)
                .build();
        }
    }
}
```

```
        ListMultipartUploadsResponse response =
s3.listMultipartUploads(listMultipartUploadsRequest);
        List<MultipartUpload> uploads = response.uploads();
        for (MultipartUpload upload : uploads) {
            System.out.println("Upload in progress: Key = \"" + upload.key() +
"\", id = " + upload.uploadId());
        }

    } catch (S3Exception e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
}
```

- Per i dettagli sull'API, consulta la [ListMultipartUploads](#) sezione AWS SDK for Java 2.x API Reference.

## ListObjectsV2

Il seguente esempio di codice mostra come utilizzare `ListObjectsV2`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.ListObjectsRequest;
import software.amazon.awssdk.services.s3.model.ListObjectsResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.S3Object;
import java.util.List;

/**
```



```
* Before running this Java V2 code example, set up your development
* environment, including your credentials.
*
* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
```

```
public class ListObjects {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <bucketName>\s

            Where:
                bucketName - The Amazon S3 bucket from which objects are read.\s
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        listBucketObjects(s3, bucketName);
        s3.close();
    }

    public static void listBucketObjects(S3Client s3, String bucketName) {
        try {
            ListObjectsRequest listObjects = ListObjectsRequest
                .builder()
                .bucket(bucketName)
                .build();

            ListObjectsResponse res = s3.listObjects(listObjects);
            List<S3Object> objects = res.contents();
            for (S3Object myValue : objects) {
```

```

        System.out.print("\n The name of the key is " + myValue.key());
        System.out.print("\n The object is " + calcKb(myValue.size()) + "
KBs");
        System.out.print("\n The owner is " + myValue.owner());
    }

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

// convert bytes to kbs.
private static long calcKb(Long val) {
    return val / 1024;
}
}

```

Elenca gli oggetti usando l'impaginazione.

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Request;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.paginators.ListObjectsV2Iterable;

public class ListObjectsPaginated {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <bucketName>\s

            Where:
                bucketName - The Amazon S3 bucket from which objects are read.\s
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }
    }
}

```

```
String bucketName = args[0];
Region region = Region.US_EAST_1;
S3Client s3 = S3Client.builder()
    .region(region)
    .build();

listBucketObjects(s3, bucketName);
s3.close();
}

public static void listBucketObjects(S3Client s3, String bucketName) {
    try {
        ListObjectsV2Request listReq = ListObjectsV2Request.builder()
            .bucket(bucketName)
            .maxKeys(1)
            .build();

        ListObjectsV2Iterable listRes = s3.listObjectsV2Paginator(listReq);
        listRes.stream()
            .flatMap(r -> r.contents().stream())
            .forEach(content -> System.out.println(" Key: " + content.key()
+ " size = " + content.size()));

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Per i dettagli sull'API, consulta la [ListObjectsversione V2](#) in AWS SDK for Java 2.x API Reference.

## PutBucketAcl

Il seguente esempio di codice mostra come utilizzarePutBucketAcl.

## SDK per Java 2.x

 Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.AccessControlPolicy;
import software.amazon.awssdk.services.s3.model.Grant;
import software.amazon.awssdk.services.s3.model.Permission;
import software.amazon.awssdk.services.s3.model.PutBucketAclRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.Type;

import java.util.ArrayList;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class SetAcl {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <bucketName> <id>\s

                Where:
                bucketName - The Amazon S3 bucket to grant permissions on.\s
                id - The ID of the owner of this bucket (you can get this value
from the AWS Management Console).
                """;

        if (args.length != 2) {
```

```
        System.out.println(usage);
        System.exit(1);
    }

    String bucketName = args[0];
    String id = args[1];
    System.out.format("Setting access \n");
    System.out.println(" in bucket: " + bucketName);
    Region region = Region.US_EAST_1;
    S3Client s3 = S3Client.builder()
        .region(region)
        .build();

    setBucketAcl(s3, bucketName, id);
    System.out.println("Done!");
    s3.close();
}

public static void setBucketAcl(S3Client s3, String bucketName, String id) {
    try {
        Grant ownerGrant = Grant.builder()
            .grantee(builder -> builder.id(id)
                .type(Type.CANONICAL_USER))
            .permission(Permission.FULL_CONTROL)
            .build();

        List<Grant> grantList2 = new ArrayList<>();
        grantList2.add(ownerGrant);

        AccessControlPolicy acl = AccessControlPolicy.builder()
            .owner(builder -> builder.id(id))
            .grants(grantList2)
            .build();

        PutBucketAclRequest putAclReq = PutBucketAclRequest.builder()
            .bucket(bucketName)
            .accessControlPolicy(acl)
            .build();

        s3.putBucketAcl(putAclReq);

    } catch (S3Exception e) {
        e.printStackTrace();
        System.exit(1);
    }
}
```

```
    }  
  }  
}
```

- Per i dettagli sull'API, consulta la [PutBucketAcl](#) sezione AWS SDK for Java 2.x API Reference.

## PutBucketCors

Il seguente esempio di codice mostra come utilizzare `PutBucketCors`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;  
import software.amazon.awssdk.services.s3.S3Client;  
import java.util.ArrayList;  
import java.util.List;  
import software.amazon.awssdk.services.s3.model.GetBucketCorsRequest;  
import software.amazon.awssdk.services.s3.model.GetBucketCorsResponse;  
import software.amazon.awssdk.services.s3.model.DeleteBucketCorsRequest;  
import software.amazon.awssdk.services.s3.model.S3Exception;  
import software.amazon.awssdk.services.s3.model.CORSRule;  
import software.amazon.awssdk.services.s3.model.CORSConfiguration;  
import software.amazon.awssdk.services.s3.model.PutBucketCorsRequest;  
  
/**  
 * Before running this Java V2 code example, set up your development  
 * environment, including your credentials.  
 *  
 * For more information, see the following documentation topic:  
 *  
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html  
 */  
public class S3Cors {  
    public static void main(String[] args) {  
        final String usage = ""
```

```

Usage:
    <bucketName> <accountId>\s

Where:
    bucketName - The Amazon S3 bucket to upload an object into.
    accountId - The id of the account that owns the Amazon S3
bucket.

    """;

if (args.length != 2) {
    System.out.println(usage);
    System.exit(1);
}

String bucketName = args[0];
String accountId = args[1];
Region region = Region.US_EAST_1;
S3Client s3 = S3Client.builder()
    .region(region)
    .build();

setCorsInformation(s3, bucketName, accountId);
getBucketCorsInformation(s3, bucketName, accountId);
deleteBucketCorsInformation(s3, bucketName, accountId);
s3.close();
}

public static void deleteBucketCorsInformation(S3Client s3, String bucketName,
String accountId) {
    try {
        DeleteBucketCorsRequest bucketCorsRequest =
DeleteBucketCorsRequest.builder()
            .bucket(bucketName)
            .expectedBucketOwner(accountId)
            .build();

        s3.deleteBucketCors(bucketCorsRequest);

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}

```

```
public static void getBucketCorsInformation(S3Client s3, String bucketName,
String accountId) {
    try {
        GetBucketCorsRequest bucketCorsRequest = GetBucketCorsRequest.builder()
            .bucket(bucketName)
            .expectedBucketOwner(accountId)
            .build();

        GetBucketCorsResponse corsResponse =
s3.getBucketCors(bucketCorsRequest);
        List<CORSRule> corsRules = corsResponse.corsRules();
        for (CORSRule rule : corsRules) {
            System.out.println("allowOrigins: " + rule.allowedOrigins());
            System.out.println("AllowedMethod: " + rule.allowedMethods());
        }

    } catch (S3Exception e) {

        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void setCorsInformation(S3Client s3, String bucketName, String
accountId) {
    List<String> allowMethods = new ArrayList<>();
    allowMethods.add("PUT");
    allowMethods.add("POST");
    allowMethods.add("DELETE");

    List<String> allowOrigins = new ArrayList<>();
    allowOrigins.add("http://example.com");
    try {
        // Define CORS rules.
        CORSRule corsRule = CORSRule.builder()
            .allowedMethods(allowMethods)
            .allowedOrigins(allowOrigins)
            .build();

        List<CORSRule> corsRules = new ArrayList<>();
        corsRules.add(corsRule);
        CORSConfiguration configuration = CORSConfiguration.builder()
            .corsRules(corsRules)
```



```
        .build();

        PutBucketCorsRequest putBucketCorsRequest =
PutBucketCorsRequest.builder()
        .bucket(bucketName)
        .corsConfiguration(configuration)
        .expectedBucketOwner(accountId)
        .build();

        s3.putBucketCors(putBucketCorsRequest);

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Per i dettagli sull'API, consulta la [PutBucketCors](#) sezione AWS SDK for Java 2.x API Reference.

## PutBucketLifecycleConfiguration

Il seguente esempio di codice mostra come utilizzare `PutBucketLifecycleConfiguration`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.LifecycleRuleFilter;
import software.amazon.awssdk.services.s3.model.Transition;
import
    software.amazon.awssdk.services.s3.model.GetBucketLifecycleConfigurationRequest;
import
    software.amazon.awssdk.services.s3.model.GetBucketLifecycleConfigurationResponse;
import software.amazon.awssdk.services.s3.model.DeleteBucketLifecycleRequest;
```

```
import software.amazon.awssdk.services.s3.model.TransitionStorageClass;
import software.amazon.awssdk.services.s3.model.LifecycleRule;
import software.amazon.awssdk.services.s3.model.ExpirationStatus;
import software.amazon.awssdk.services.s3.model.BucketLifecycleConfiguration;
import
    software.amazon.awssdk.services.s3.model.PutBucketLifecycleConfigurationRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import java.util.ArrayList;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class LifecycleConfiguration {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <bucketName> <accountId>\s

            Where:
                bucketName - The Amazon Simple Storage Service
(Amazon S3) bucket to upload an object into.
                accountId - The id of the account that owns the
Amazon S3 bucket.

            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String accountId = args[1];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();
    }
}
```

```
        setLifecycleConfig(s3, bucketName, accountId);
        getLifecycleConfig(s3, bucketName, accountId);
        deleteLifecycleConfig(s3, bucketName, accountId);
        System.out.println("You have successfully created, updated, and
deleted a Lifecycle configuration");
        s3.close();
    }

    public static void setLifecycleConfig(S3Client s3, String bucketName, String
accountId) {
        try {
            // Create a rule to archive objects with the
"glacierobjects/" prefix to Amazon
            // S3 Glacier.
            LifecycleRuleFilter ruleFilter =
LifecycleRuleFilter.builder()
                .prefix("glacierobjects/")
                .build();

            Transition transition = Transition.builder()

.storageClass(TransitionStorageClass.GLACIER)
                .days(0)
                .build();

            LifecycleRule rule1 = LifecycleRule.builder()
                .id("Archive immediately rule")
                .filter(ruleFilter)
                .transitions(transition)
                .status(ExpirationStatus.ENABLED)
                .build();

            // Create a second rule.
            Transition transition2 = Transition.builder()

.storageClass(TransitionStorageClass.GLACIER)
                .days(0)
                .build();

            List<Transition> transitionList = new ArrayList<>();
            transitionList.add(transition2);
```

```
        LifecycleRuleFilter ruleFilter2 =
LifecycleRuleFilter.builder()
                        .prefix("glacierobjects/")
                        .build();

        LifecycleRule rule2 = LifecycleRule.builder()
                        .id("Archive and then delete rule")
                        .filter(ruleFilter2)
                        .transitions(transitionList)
                        .status(ExpirationStatus.ENABLED)
                        .build();

        // Add the LifecycleRule objects to an ArrayList.
        ArrayList<LifecycleRule> ruleList = new ArrayList<>();
        ruleList.add(rule1);
        ruleList.add(rule2);

        BucketLifecycleConfiguration lifecycleConfiguration =
BucketLifecycleConfiguration.builder()
                        .rules(ruleList)
                        .build();

        PutBucketLifecycleConfigurationRequest
putBucketLifecycleConfigurationRequest = PutBucketLifecycleConfigurationRequest
                        .builder()
                        .bucket(bucketName)

.lifecycleConfiguration(lifecycleConfiguration)
                        .expectedBucketOwner(accountId)
                        .build();

s3.putBucketLifecycleConfiguration(putBucketLifecycleConfigurationRequest);

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    // Retrieve the configuration and add a new rule.
    public static void getLifecycleConfig(S3Client s3, String bucketName, String
accountId) {
        try {
```

```
GetBucketLifecycleConfigurationRequest
getBucketLifecycleConfigurationRequest = GetBucketLifecycleConfigurationRequest
    .builder()
    .bucket(bucketName)
    .expectedBucketOwner(accountId)
    .build();

GetBucketLifecycleConfigurationResponse response = s3
    .getBucketLifecycleConfiguration(getBucketLifecycleConfigurationRequest);
List<LifecycleRule> newList = new ArrayList<>();
List<LifecycleRule> rules = response.rules();
for (LifecycleRule rule : rules) {
    newList.add(rule);
}

// Add a new rule with both a prefix predicate and a tag
predicate.
LifecycleRuleFilter ruleFilter =
LifecycleRuleFilter.builder()
    .prefix("YearlyDocuments/")
    .build();

Transition transition = Transition.builder()
    .storageClass(TransitionStorageClass.GLACIER)
    .days(3650)
    .build();

LifecycleRule rule1 = LifecycleRule.builder()
    .id("NewRule")
    .filter(ruleFilter)
    .transitions(transition)
    .status(ExpirationStatus.ENABLED)
    .build();

// Add the new rule to the list.
newList.add(rule1);
BucketLifecycleConfiguration lifecycleConfiguration =
BucketLifecycleConfiguration.builder()
    .rules(newList)
    .build();
```

```
        PutBucketLifecycleConfigurationRequest
putBucketLifecycleConfigurationRequest = PutBucketLifecycleConfigurationRequest
        .builder()
        .bucket(bucketName)

        .lifecycleConfiguration(lifecycleConfiguration)
        .expectedBucketOwner(accountId)
        .build();

s3.putBucketLifecycleConfiguration(putBucketLifecycleConfigurationRequest);

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    // Delete the configuration from the Amazon S3 bucket.
    public static void deleteLifecycleConfig(S3Client s3, String bucketName,
String accountId) {
        try {
            DeleteBucketLifecycleRequest deleteBucketLifecycleRequest =
DeleteBucketLifecycleRequest

                .builder()
                .bucket(bucketName)
                .expectedBucketOwner(accountId)
                .build();

            s3.deleteBucketLifecycle(deleteBucketLifecycleRequest);

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Per i dettagli sull'API, consulta la [PutBucketLifecycleConfiguration](#) sezione AWS SDK for Java 2.x API Reference.

## PutBucketNotificationConfiguration

Il seguente esempio di codice mostra come utilizzare `PutBucketNotificationConfiguration`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.Event;
import software.amazon.awssdk.services.s3.model.NotificationConfiguration;
import
    software.amazon.awssdk.services.s3.model.PutBucketNotificationConfigurationRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.TopicConfiguration;
import java.util.ArrayList;
import java.util.List;

public class SetBucketEventBridgeNotification {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <bucketName>\s

                Where:
                bucketName - The Amazon S3 bucket.\s
                topicArn - The Simple Notification Service topic ARN.\s
                id - An id value used for the topic configuration. This value is
displayed in the AWS Management Console.\s
                """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
```

```
String topicArn = args[1];
String id = args[2];
Region region = Region.US_EAST_1;
S3Client s3Client = S3Client.builder()
    .region(region)
    .build();

setBucketNotification(s3Client, bucketName, topicArn, id);
s3Client.close();
}

public static void setBucketNotification(S3Client s3Client, String bucketName,
String topicArn, String id) {
    try {
        List<Event> events = new ArrayList<>();
        events.add(Event.S3_OBJECT_CREATED_PUT);

        TopicConfiguration config = TopicConfiguration.builder()
            .topicArn(topicArn)
            .events(events)
            .id(id)
            .build();

        List<TopicConfiguration> topics = new ArrayList<>();
        topics.add(config);

        NotificationConfiguration configuration =
NotificationConfiguration.builder()
            .topicConfigurations(topics)
            .build();

        PutBucketNotificationConfigurationRequest configurationRequest =
PutBucketNotificationConfigurationRequest
            .builder()
            .bucket(bucketName)
            .notificationConfiguration(configuration)
            .skipDestinationValidation(true)
            .build();

        // Set the bucket notification configuration.
        s3Client.putBucketNotificationConfiguration(configurationRequest);
        System.out.println("Added bucket " + bucketName + " with EventBridge
events enabled.");
    }
}
```



```

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}

```

- Per i dettagli sull'API, consulta la [PutBucketNotificationConfiguration](#) sezione AWS SDK for Java 2.x API Reference.

## PutBucketPolicy

Il seguente esempio di codice mostra come utilizzare PutBucketPolicy.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```

import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutBucketPolicyRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.regions.Region;
import java.io.IOException;
import java.nio.charset.StandardCharsets;
import java.nio.file.Files;
import java.nio.file.Paths;
import java.util.List;
import com.fasterxml.jackson.core.JsonParser;
import com.fasterxml.jackson.databind.ObjectMapper;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 */

```

```

* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class SetBucketPolicy {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <bucketName> <polFile>

                Where:
                bucketName - The Amazon S3 bucket to set the policy on.
                polFile - A JSON file containing the policy (see the Amazon S3
Readme for an example).\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String polFile = args[1];
        String policyText = getBucketPolicyFromFile(polFile);
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        setPolicy(s3, bucketName, policyText);
        s3.close();
    }

    public static void setPolicy(S3Client s3, String bucketName, String policyText)
    {
        System.out.println("Setting policy:");
        System.out.println("----");
        System.out.println(policyText);
        System.out.println("----");
        System.out.format("On Amazon S3 bucket: \"%s\"\n", bucketName);

        try {
            PutBucketPolicyRequest policyReq = PutBucketPolicyRequest.builder()
                .bucket(bucketName)
                .policy(policyText)

```

```
        .build();

        s3.putBucketPolicy(policyReq);

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }

    System.out.println("Done!");
}

// Loads a JSON-formatted policy from a file
public static String getBucketPolicyFromFile(String policyFile) {

    StringBuilder fileText = new StringBuilder();
    try {
        List<String> lines = Files.readAllLines(Paths.get(policyFile),
StandardCharsets.UTF_8);
        for (String line : lines) {
            fileText.append(line);
        }

    } catch (IOException e) {
        System.out.format("Problem reading file: \"%s\"", policyFile);
        System.out.println(e.getMessage());
    }

    try {
        final JsonParser parser = new
ObjectMapper().getFactory().createParser(fileText.toString());
        while (parser.nextToken() != null) {
        }

    } catch (IOException jpe) {
        jpe.printStackTrace();
    }
    return fileText.toString();
}
}
```

- Per i dettagli sull'API, consulta la [PutBucketPolicy](#) sezione AWS SDK for Java 2.x API Reference.

## PutBucketWebsite

Il seguente esempio di codice mostra come utilizzare `PutBucketWebsite`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.IndexDocument;
import software.amazon.awssdk.services.s3.model.PutBucketWebsiteRequest;
import software.amazon.awssdk.services.s3.model.WebsiteConfiguration;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.regions.Region;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class SetWebsiteConfiguration {
    public static void main(String[] args) {
        final String usage = ""

                Usage:    <bucketName> [indexdoc]\s

                Where:
                bucketName    - The Amazon S3 bucket to set the website
configuration on.\s
                indexdoc - The index document, ex. 'index.html'
                        If not specified, 'index.html' will be set.
    }
}
```

```
        """;

    if (args.length != 1) {
        System.out.println(usage);
        System.exit(1);
    }

    String bucketName = args[0];
    String indexDoc = "index.html";
    Region region = Region.US_EAST_1;
    S3Client s3 = S3Client.builder()
        .region(region)
        .build();

    setWebsiteConfig(s3, bucketName, indexDoc);
    s3.close();
}

public static void setWebsiteConfig(S3Client s3, String bucketName, String
indexDoc) {
    try {
        WebsiteConfiguration websiteConfig = WebsiteConfiguration.builder()
            .indexDocument(IndexDocument.builder().suffix(indexDoc).build())
            .build();

        PutBucketWebsiteRequest pubWebsiteReq =
PutBucketWebsiteRequest.builder()
            .bucket(bucketName)
            .websiteConfiguration(websiteConfig)
            .build();

        s3.putBucketWebsite(pubWebsiteReq);
        System.out.println("The call was successful");

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Per i dettagli sull'API, consulta la [PutBucketWebsite](#) sezione AWS SDK for Java 2.x API Reference.

## PutObject

Il seguente esempio di codice mostra come utilizzare `PutObject`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Carica un file in un bucket utilizzando un'interfaccia [S3Client](#).

```
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import java.io.File;
import java.util.HashMap;
import java.util.Map;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class PutObject {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <bucketName> <objectKey> <objectPath>\s
```

```

        Where:
        bucketName - The Amazon S3 bucket to upload an object into.
        objectKey - The object to upload (for example, book.pdf).
        objectPath - The path where the file is located (for example, C:/
AWS/book2.pdf).\s
        """;

    if (args.length != 3) {
        System.out.println(usage);
        System.exit(1);
    }

    String bucketName = args[0];
    String objectKey = args[1];
    String objectPath = args[2];
    Region region = Region.US_EAST_1;
    S3Client s3 = S3Client.builder()
        .region(region)
        .build();

    putS3Object(s3, bucketName, objectKey, objectPath);
    s3.close();
}

// This example uses RequestBody.fromFile to avoid loading the whole file into
// memory.
public static void putS3Object(S3Client s3, String bucketName, String objectKey,
String objectPath) {
    try {
        Map<String, String> metadata = new HashMap<>();
        metadata.put("x-amz-meta-myVal", "test");
        PutObjectRequest putOb = PutObjectRequest.builder()
            .bucket(bucketName)
            .key(objectKey)
            .metadata(metadata)
            .build();

        s3.putObject(putOb, RequestBody.fromFile(new File(objectPath)));
        System.out.println("Successfully placed " + objectKey + " into bucket "
+ bucketName);

    } catch (S3Exception e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

```

```
    }  
  }  
}
```

Usa un [S3 TransferManager](#) per [caricare un file](#) in un bucket. Visualizza il [file completo](#) ed esegui il [test](#).

```
import org.slf4j.Logger;  
import org.slf4j.LoggerFactory;  
import software.amazon.awssdk.transfer.s3.S3TransferManager;  
import software.amazon.awssdk.transfer.s3.model.CompletedFileUpload;  
import software.amazon.awssdk.transfer.s3.model.FileUpload;  
import software.amazon.awssdk.transfer.s3.model.UploadFileRequest;  
import software.amazon.awssdk.transfer.s3.progress.LoggingTransferListener;  
import java.net.URI;  
import java.net.URISyntaxException;  
import java.net.URL;  
import java.nio.file.Paths;  
import java.util.UUID;  
  
    public String uploadFile(S3TransferManager transferManager, String bucketName,  
                            String key, URI filePathURI) {  
        UploadFileRequest uploadFileRequest = UploadFileRequest.builder()  
            .putObjectRequest(b -> b.bucket(bucketName).key(key))  
            .source(Paths.get(filePathURI))  
            .build();  
  
        FileUpload fileUpload = transferManager.uploadFile(uploadFileRequest);  
  
        CompletedFileUpload uploadResult = fileUpload.completionFuture().join();  
        return uploadResult.response().eTag();  
    }
```

Caricare un oggetto in un bucket e impostare tag mediante un'interfaccia [S3Client](#).

```
    public static void putS3ObjectTags(S3Client s3, String bucketName, String  
objectKey, String objectPath) {  
    try {  
        Tag tag1 = Tag.builder()  
            .key("Tag 1")  
            .value("This is tag 1")
```



```
        .build();

    Tag tag2 = Tag.builder()
        .key("Tag 2")
        .value("This is tag 2")
        .build();

    List<Tag> tags = new ArrayList<>();
    tags.add(tag1);
    tags.add(tag2);

    Tagging allTags = Tagging.builder()
        .tagSet(tags)
        .build();

    PutObjectRequest putOb = PutObjectRequest.builder()
        .bucket(bucketName)
        .key(objectKey)
        .tagging(allTags)
        .build();

    s3.putObject(putOb, RequestBody.fromBytes(getObjectFile(objectPath)));

} catch (S3Exception e) {
    System.err.println(e.getMessage());
    System.exit(1);
}

}

public static void updateObjectTags(S3Client s3, String bucketName, String
objectKey) {
    try {
        GetObjectTaggingRequest taggingRequest =
GetObjectTaggingRequest.builder()
            .bucket(bucketName)
            .key(objectKey)
            .build();

        GetObjectTaggingResponse getTaggingRes =
s3.getObjectTagging(taggingRequest);
        List<Tag> obTags = getTaggingRes.tagSet();
        for (Tag sinTag : obTags) {
            System.out.println("The tag key is: " + sinTag.key());
            System.out.println("The tag value is: " + sinTag.value());
        }
    }
}
```

```
    }

    // Replace the object's tags with two new tags.
    Tag tag3 = Tag.builder()
        .key("Tag 3")
        .value("This is tag 3")
        .build();

    Tag tag4 = Tag.builder()
        .key("Tag 4")
        .value("This is tag 4")
        .build();

    List<Tag> tags = new ArrayList<>();
    tags.add(tag3);
    tags.add(tag4);

    Tagging updatedTags = Tagging.builder()
        .tagSet(tags)
        .build();

    PutObjectTaggingRequest taggingRequest1 =
    PutObjectTaggingRequest.builder()
        .bucket(bucketName)
        .key(objectKey)
        .tagging(updatedTags)
        .build();

    s3.putObjectTagging(taggingRequest1);
    GetObjectTaggingResponse getTaggingRes2 =
    s3.getObjectTagging(taggingRequest);
    List<Tag> modTags = getTaggingRes2.tagSet();
    for (Tag sinTag : modTags) {
        System.out.println("The tag key is: " + sinTag.key());
        System.out.println("The tag value is: " + sinTag.value());
    }

    } catch (S3Exception e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

// Return a byte array.
```

```

private static byte[] getObjectFile(String filePath) {
    FileInputStream fileInputStream = null;
    byte[] byteArray = null;

    try {
        File file = new File(filePath);
        byteArray = new byte[(int) file.length()];
        fileInputStream = new FileInputStream(file);
        fileInputStream.read(byteArray);

    } catch (IOException e) {
        e.printStackTrace();
    } finally {
        if (fileInputStream != null) {
            try {
                fileInputStream.close();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }

    return byteArray;
}
}

```

Caricare un oggetto in un bucket e impostare metadati mediante un'interfaccia [S3Client](#).

```

import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import java.io.File;
import java.util.HashMap;
import java.util.Map;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:

```

```

*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class PutObjectMetadata {
    public static void main(String[] args) {
        final String USAGE = ""

            Usage:
                <bucketName> <objectKey> <objectPath>\s

            Where:
                bucketName - The Amazon S3 bucket to upload an object into.
                objectKey - The object to upload (for example, book.pdf).
                objectPath - The path where the file is located (for example, C:/
AWS/book2.pdf).\s
                """;

        if (args.length != 3) {
            System.out.println(USAGE);
            System.exit(1);
        }

        String bucketName = args[0];
        String objectKey = args[1];
        String objectPath = args[2];
        System.out.println("Putting object " + objectKey + " into bucket " +
bucketName);
        System.out.println("  in bucket: " + bucketName);
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        putS3Object(s3, bucketName, objectKey, objectPath);
        s3.close();
    }

    // This example uses RequestBody.fromFile to avoid loading the whole file into
    // memory.
    public static void putS3Object(S3Client s3, String bucketName, String objectKey,
String objectPath) {
        try {
            Map<String, String> metadata = new HashMap<>();
            metadata.put("author", "Mary Doe");

```

```

        metadata.put("version", "1.0.0.0");

        PutObjectRequest putOb = PutObjectRequest.builder()
            .bucket(bucketName)
            .key(objectKey)
            .metadata(metadata)
            .build();

        s3.putObject(putOb, RequestBody.fromFile(new File(objectPath)));
        System.out.println("Successfully placed " + objectKey + " into bucket "
+ bucketName);

    } catch (S3Exception e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
}

```

Caricare un oggetto in un bucket e impostare un valore di conservazione per l'oggetto mediante un'interfaccia [S3Client](#).

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutObjectRetentionRequest;
import software.amazon.awssdk.services.s3.model.ObjectLockRetention;
import software.amazon.awssdk.services.s3.model.S3Exception;
import java.time.Instant;
import java.time.LocalDate;
import java.time.LocalDateTime;
import java.time.ZoneOffset;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class PutObjectRetention {

```

```
public static void main(String[] args) {
    final String usage = ""

        Usage:
            <key> <bucketName>\s

        Where:
            key - The name of the object (for example, book.pdf).\s
            bucketName - The Amazon S3 bucket name that contains the object
(for example, bucket1).\s
        """;

    if (args.length != 2) {
        System.out.println(usage);
        System.exit(1);
    }

    String key = args[0];
    String bucketName = args[1];
    Region region = Region.US_EAST_1;
    S3Client s3 = S3Client.builder()
        .region(region)
        .build();

    setRetentionPeriod(s3, key, bucketName);
    s3.close();
}

public static void setRetentionPeriod(S3Client s3, String key, String bucket) {
    try {
        LocalDate localDate = LocalDate.parse("2020-07-17");
        LocalDateTime localDateTime = localDate.atStartOfDay();
        Instant instant = localDateTime.toInstant(ZoneOffset.UTC);

        ObjectLockRetention lockRetention = ObjectLockRetention.builder()
            .mode("COMPLIANCE")
            .retainUntilDate(instant)
            .build();

        PutObjectRetentionRequest retentionRequest =
PutObjectRetentionRequest.builder()
            .bucket(bucket)
            .key(key)
            .bypassGovernanceRetention(true)
```

```

        .retention(lockRetention)
        .build();

    // To set Retention on an object, the Amazon S3 bucket must support
object
    // locking, otherwise an exception is thrown.
    s3.putObjectRetention(retentionRequest);
    System.out.print("An object retention configuration was successfully
placed on the object");

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
}

```

- Per i dettagli sull'API, consulta la sezione AWS SDK for Java 2.x API [PutObjectReference](#).

## PutObjectLegalHold

Il seguente esempio di codice mostra come utilizzare `PutObjectLegalHold`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```

// Set or modify a legal hold on an object in an S3 bucket.
public void modifyObjectLegalHold(String bucketName, String objectKey, boolean
legalHoldOn) {
    ObjectLockLegalHold legalHold ;
    if (legalHoldOn) {
        legalHold = ObjectLockLegalHold.builder()
            .status(ObjectLockLegalHoldStatus.ON)
            .build();
    } else {
        legalHold = ObjectLockLegalHold.builder()

```

```
        .status(ObjectLockLegalHoldStatus.OFF)
        .build();
    }

    PutObjectLegalHoldRequest legalHoldRequest =
PutObjectLegalHoldRequest.builder()
        .bucket(bucketName)
        .key(objectKey)
        .legalHold(legalHold)
        .build();

    getClient().putObjectLegalHold(legalHoldRequest) ;
    System.out.println("Modified legal hold for "+ objectKey +" in "+bucketName
+".");
}
```

- Per i dettagli sull'API, consulta la [PutObjectLegalHold](#) sezione AWS SDK for Java 2.x API Reference.

## PutObjectLockConfiguration

Il seguente esempio di codice mostra come utilizzare `PutObjectLockConfiguration`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Imposta la configurazione di blocco degli oggetti di un bucket.

```
// Enable object lock on an existing bucket.
public void enableObjectLockOnBucket(String bucketName) {
    try {
        VersioningConfiguration versioningConfiguration =
VersioningConfiguration.builder()
            .status(BucketVersioningStatus.ENABLED)
            .build();
```



```

        PutBucketVersioningRequest putBucketVersioningRequest =
PutBucketVersioningRequest.builder()
    .bucket(bucketName)
    .versioningConfiguration(versioningConfiguration)
    .build();

    // Enable versioning on the bucket.
    getClient().putBucketVersioning(putBucketVersioningRequest);
    PutObjectLockConfigurationRequest request =
PutObjectLockConfigurationRequest.builder()
    .bucket(bucketName)
    .objectLockConfiguration(ObjectLockConfiguration.builder()
        .objectLockEnabled(ObjectLockEnabled.ENABLED)
        .build())
    .build();

    getClient().putObjectLockConfiguration(request);
    System.out.println("Successfully enabled object lock on "+bucketName);

} catch (S3Exception ex) {
    System.out.println("Error modifying object lock: '" + ex.getMessage() +
    "'");
}
}

```

Imposta il periodo di conservazione predefinito di un bucket.

```

// Set or modify a retention period on an S3 bucket.
public void modifyBucketDefaultRetention(String bucketName) {
    VersioningConfiguration versioningConfiguration =
VersioningConfiguration.builder()
    .mfaDelete(MFADelete.DISABLED)
    .status(BucketVersioningStatus.ENABLED)
    .build();

    PutBucketVersioningRequest versioningRequest =
PutBucketVersioningRequest.builder()
    .bucket(bucketName)
    .versioningConfiguration(versioningConfiguration)
    .build();

    getClient().putBucketVersioning(versioningRequest);
}

```

```
DefaultRetention retention = DefaultRetention.builder()
    .days(1)
    .mode(ObjectLockRetentionMode.GOVERNANCE)
    .build();

ObjectLockRule lockRule = ObjectLockRule.builder()
    .defaultRetention(retention)
    .build();

ObjectLockConfiguration objectLockConfiguration =
ObjectLockConfiguration.builder()
    .objectLockEnabled(ObjectLockEnabled.ENABLED)
    .rule(lockRule)
    .build();

PutObjectLockConfigurationRequest putObjectLockConfigurationRequest =
PutObjectLockConfigurationRequest.builder()
    .bucket(bucketName)
    .objectLockConfiguration(objectLockConfiguration)
    .build();

getClient().putObjectLockConfiguration(putObjectLockConfigurationRequest) ;
System.out.println("Added a default retention to bucket "+bucketName +".");
}
```

- Per i dettagli sull'API, consulta la sezione [PutObjectLockConfiguration AWS SDK for Java 2.x API Reference](#).

## PutObjectRetention

Il seguente esempio di codice mostra come utilizzare PutObjectRetention.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
// Set or modify a retention period on an object in an S3 bucket.
public void modifyObjectRetentionPeriod(String bucketName, String objectKey) {
    // Calculate the instant one day from now.
    Instant futureInstant = Instant.now().plus(1, ChronoUnit.DAYS);

    // Convert the Instant to a ZonedDateTime object with a specific time zone.
    ZonedDateTime zonedDateTime = futureInstant.atZone(ZoneId.systemDefault());

    // Define a formatter for human-readable output.
    DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd
HH:mm:ss");

    // Format the ZonedDateTime object to a human-readable date string.
    String humanReadableDate = formatter.format(zonedDateTime);

    // Print the formatted date string.
    System.out.println("Formatted Date: " + humanReadableDate);
    ObjectLockRetention retention = ObjectLockRetention.builder()
        .mode(ObjectLockRetentionMode.GOVERNANCE)
        .retainUntilDate(futureInstant)
        .build();

    PutObjectRetentionRequest retentionRequest =
PutObjectRetentionRequest.builder()
    .bucket(bucketName)
    .key(objectKey)
    .retention(retention)
    .build();

    getClient().putObjectRetention(retentionRequest);
    System.out.println("Set retention for "+objectKey +" in " +bucketName +"
until "+ humanReadableDate +".");
}
```

- Per i dettagli sull'API, consulta la [PutObjectRetention](#) sezione AWS SDK for Java 2.x API Reference.

## RestoreObject

Il seguente esempio di codice mostra come utilizzare `RestoreObject`.

## SDK per Java 2.x

 Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.RestoreRequest;
import software.amazon.awssdk.services.s3.model.GlacierJobParameters;
import software.amazon.awssdk.services.s3.model.RestoreObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.Tier;

/*
 * For more information about restoring an object, see "Restoring an archived
 * object" at
 * https://docs.aws.amazon.com/AmazonS3/latest/userguide/restoring-objects.html
 *
 * Before running this Java V2 code example, set up your development environment,
 * including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class RestoreObject {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <bucketName> <keyName> <expectedBucketOwner>

                Where:
                bucketName - The Amazon S3 bucket name.\s
                keyName - The key name of an object with a Storage class value
of Glacier.\s
                expectedBucketOwner - The account that owns the bucket (you can
obtain this value from the AWS Management Console).\s
                """;
```

```
    if (args.length != 3) {
        System.out.println(usage);
        System.exit(1);
    }

    String bucketName = args[0];
    String keyName = args[1];
    String expectedBucketOwner = args[2];
    Region region = Region.US_EAST_1;
    S3Client s3 = S3Client.builder()
        .region(region)
        .build();

    restoreS3Object(s3, bucketName, keyName, expectedBucketOwner);
    s3.close();
}

public static void restoreS3Object(S3Client s3, String bucketName, String
keyName, String expectedBucketOwner) {
    try {
        RestoreRequest restoreRequest = RestoreRequest.builder()
            .days(10)

.glacierJobParameters(GlacierJobParameters.builder().tier(Tier.STANDARD).build())
            .build();

        RestoreObjectRequest objectRequest = RestoreObjectRequest.builder()
            .expectedBucketOwner(expectedBucketOwner)
            .bucket(bucketName)
            .key(keyName)
            .restoreRequest(restoreRequest)
            .build();

        s3.restoreObject(objectRequest);

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Per i dettagli sull'API, consulta la [RestoreObject](#) sezione AWS SDK for Java 2.x API Reference.

## SelectObjectContent

Il seguente esempio di codice mostra come utilizzare `SelectObjectContent`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

L'esempio seguente mostra una query che utilizza un oggetto JSON. L'[esempio completo](#) mostra anche l'uso di un oggetto CSV.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.async.AsyncRequestBody;
import software.amazon.awssdk.core.async.BlockingInputStreamAsyncRequestBody;
import software.amazon.awssdk.core.exception.SdkException;
import software.amazon.awssdk.services.s3.S3AsyncClient;
import software.amazon.awssdk.services.s3.model.CSVInput;
import software.amazon.awssdk.services.s3.model.CSVOutput;
import software.amazon.awssdk.services.s3.model.CompressionType;
import software.amazon.awssdk.services.s3.model.ExpressionType;
import software.amazon.awssdk.services.s3.model.FileHeaderInfo;
import software.amazon.awssdk.services.s3.model.InputSerialization;
import software.amazon.awssdk.services.s3.model.JSONInput;
import software.amazon.awssdk.services.s3.model.JSONOutput;
import software.amazon.awssdk.services.s3.model.JSONType;
import software.amazon.awssdk.services.s3.model.ObjectIdentifier;
import software.amazon.awssdk.services.s3.model.OutputSerialization;
import software.amazon.awssdk.services.s3.model.Progress;
import software.amazon.awssdk.services.s3.model.PutObjectResponse;
import software.amazon.awssdk.services.s3.model.SelectObjectContentRequest;
import software.amazon.awssdk.services.s3.model.SelectObjectContentResponseHandler;
import software.amazon.awssdk.services.s3.model.Stats;

import java.io.IOException;
```

```
import java.net.URL;
import java.util.ArrayList;
import java.util.List;
import java.util.UUID;
import java.util.concurrent.CompletableFuture;

public class SelectObjectContentExample {
    static final Logger logger =
    LoggerFactory.getLogger(SelectObjectContentExample.class);
    static final String BUCKET_NAME = "select-object-content-" + UUID.randomUUID();
    static final S3AsyncClient s3AsyncClient = S3AsyncClient.create();
    static String FILE_CSV = "csv";
    static String FILE_JSON = "json";
    static String URL_CSV = "https://raw.githubusercontent.com/mledoze/countries/
master/dist/countries.csv";
    static String URL_JSON = "https://raw.githubusercontent.com/mledoze/countries/
master/dist/countries.json";

    public static void main(String[] args) {
        SelectObjectContentExample selectObjectContentExample = new
        SelectObjectContentExample();
        try {
            SelectObjectContentExample.setUp();
            selectObjectContentExample.runSelectObjectContentMethodForJSON();
            selectObjectContentExample.runSelectObjectContentMethodForCSV();
        } catch (SdkException e) {
            logger.error(e.getMessage(), e);
            System.exit(1);
        } finally {
            SelectObjectContentExample.tearDown();
        }
    }

    EventStreamInfo runSelectObjectContentMethodForJSON() {
        // Set up request parameters.
        final String queryExpression = "select * from s3object[*][*] c where c.area
< 350000";
        final String fileType = FILE_JSON;

        InputSerialization inputSerialization = InputSerialization.builder()
            .json(JSONInput.builder().type(JSONType.DOCUMENT).build())
            .compressionType(CompressionType.NONE)
            .build();
    }
}
```

```

OutputSerialization outputSerialization = OutputSerialization.builder()
    .json(JSONOutput.builder().recordDelimiter(null).build())
    .build();

// Build the SelectObjectContentRequest.
SelectObjectContentRequest select = SelectObjectContentRequest.builder()
    .bucket(BUCKET_NAME)
    .key(FILE_JSON)
    .expression(queryExpression)
    .expressionType(ExpressionType.SQL)
    .inputSerialization(inputSerialization)
    .outputSerialization(outputSerialization)
    .build();

EventStreamInfo eventStreamInfo = new EventStreamInfo();
// Call the selectObjectContent method with the request and a response
handler.
// Supply an EventStreamInfo object to the response handler to gather
records and information from the response.
s3AsyncClient.selectObjectContent(select,
buildResponseHandler(eventStreamInfo)).join();

// Log out information gathered while processing the response stream.
long recordCount = eventStreamInfo.getRecords().stream().mapToInt(record ->
    record.split("\n").length
).sum();
logger.info("Total records {}: {}", fileType, recordCount);
logger.info("Visitor onRecords for fileType {} called {} times", fileType,
eventStreamInfo.getCountOnRecordsCalled());
logger.info("Visitor onStats for fileType {}, {}", fileType,
eventStreamInfo.getStats());
logger.info("Visitor onContinuations for fileType {}, {}", fileType,
eventStreamInfo.getCountContinuationEvents());
return eventStreamInfo;
}

static SelectObjectContentResponseHandler buildResponseHandler(EventStreamInfo
eventStreamInfo) {
// Use a Visitor to process the response stream. This visitor logs
information and gathers details while processing.
final SelectObjectContentResponseHandler.Visitor visitor =
SelectObjectContentResponseHandler.Visitor.builder()
    .onRecords(r -> {
        logger.info("Record event received.");
    });
}

```



```

        eventStreamInfo.addRecord(r.payload().asUtf8String());
        eventStreamInfo.incrementOnRecordsCalled();
    })
    .onCont(ce -> {
        logger.info("Continuation event received.");
        eventStreamInfo.incrementContinuationEvents();
    })
    .onProgress(pe -> {
        Progress progress = pe.details();
        logger.info("Progress event received:\n bytesScanned:
{} \n bytesProcessed: {} \n bytesReturned: {}",
            progress.bytesScanned(),
            progress.bytesProcessed(),
            progress.bytesReturned());
    })
    .onEnd(ee -> logger.info("End event received."))
    .onStats(se -> {
        logger.info("Stats event received.");
        eventStreamInfo.addStats(se.details());
    })
    .build();

    // Build the SelectObjectContentResponseHandler with the visitor that
processes the stream.
    return SelectObjectContentResponseHandler.builder()
        .subscriber(visitor).build();
}

// The EventStreamInfo class is used to store information gathered while
processing the response stream.
static class EventStreamInfo {
    private final List<String> records = new ArrayList<>();
    private Integer countOnRecordsCalled = 0;
    private Integer countContinuationEvents = 0;
    private Stats stats;

    void incrementOnRecordsCalled() {
        countOnRecordsCalled++;
    }

    void incrementContinuationEvents() {
        countContinuationEvents++;
    }
}

```

```
    void addRecord(String record) {
        records.add(record);
    }

    void addStats(Stats stats) {
        this.stats = stats;
    }

    public List<String> getRecords() {
        return records;
    }

    public Integer getCountOnRecordsCalled() {
        return countOnRecordsCalled;
    }

    public Integer getCountContinuationEvents() {
        return countContinuationEvents;
    }

    public Stats getStats() {
        return stats;
    }
}
```

- Per i dettagli sull'API, consulta la sezione [SelectObjectContent AWS SDK for Java 2.xAPI Reference](#).

## Scenari

### Creazione di un URL prefirmato

Il seguente esempio di codice mostra come creare un URL predefinito per Amazon S3 e caricare un oggetto.

## SDK per Java 2.x

 Note

C'è altro su [GitHub](#) Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Genera un URL prefirmato per un oggetto, quindi scaricalo (richiesta GET).

## Importazioni.

```
import com.example.s3.util.PresignUrlUtils;
import org.slf4j.Logger;
import software.amazon.awssdk.http.HttpExecuteRequest;
import software.amazon.awssdk.http.HttpExecuteResponse;
import software.amazon.awssdk.http.SdkHttpClient;
import software.amazon.awssdk.http.SdkHttpMethod;
import software.amazon.awssdk.http.SdkHttpRequest;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.presigner.S3Presigner;
import software.amazon.awssdk.services.s3.presigner.model.GetObjectPresignRequest;
import software.amazon.awssdk.services.s3.presigner.model.PresignedGetObjectRequest;
import software.amazon.awssdk.utils.IoUtils;

import java.io.ByteArrayOutputStream;
import java.io.File;
import java.io.IOException;
import java.io.InputStream;
import java.net.HttpURLConnection;
import java.net.URISyntaxException;
import java.net.URL;
import java.net.http.HttpClient;
import java.net.http.HttpRequest;
import java.net.http.HttpResponse;
import java.nio.file.Paths;
import java.time.Duration;
import java.util.UUID;
```

## Genera l'URL.

```
/* Create a pre-signed URL to download an object in a subsequent GET request. */
public String createPresignedGetUrl(String bucketName, String keyName) {
    try (S3Presigner presigner = S3Presigner.create()) {

        GetObjectRequest objectRequest = GetObjectRequest.builder()
            .bucket(bucketName)
            .key(keyName)
            .build();

        GetObjectPresignRequest presignRequest =
        GetObjectPresignRequest.builder()
            .signatureDuration(Duration.ofMinutes(10)) // The URL will
            expire in 10 minutes.
            .getObjectRequest(objectRequest)
            .build();

        PresignedGetObjectRequest presignedRequest =
        presigner.presignGetObject(presignRequest);
        logger.info("Presigned URL: [{}]", presignedRequest.url().toString());
        logger.info("HTTP method: [{}]",
        presignedRequest.httpRequest().method());

        return presignedRequest.url().toExternalForm();
    }
}
```

Scaricate l'oggetto utilizzando uno dei tre approcci seguenti.

Usa la classe JDK `URLConnection` (dalla v1.1) per eseguire il download.

```
/* Use the JDK HttpURLConnection (since v1.1) class to do the download. */
public byte[] useHttpURLConnectionToGet(String presignedUrlString) {
    ByteArrayOutputStream byteArrayOutputStream = new
    ByteArrayOutputStream(); // Capture the response body to a byte array.

    try {
        URL presignedUrl = new URL(presignedUrlString);
        HttpURLConnection connection = (HttpURLConnection)
        presignedUrl.openConnection();
        connection.setRequestMethod("GET");
```

```

        // Download the result of executing the request.
        try (InputStream content = connection.getInputStream()) {
            IoUtils.copy(content, byteArrayOutputStream);
        }
        logger.info("HTTP response code is " + connection.getResponseCode());

    } catch (S3Exception | IOException e) {
        logger.error(e.getMessage(), e);
    }
    return byteArrayOutputStream.toByteArray();
}

```

Usa la classe JDK `HttpClient` (dalla v11) per eseguire il download.

```

/* Use the JDK HttpClient (since v11) class to do the download. */
public byte[] useHttpClientToGet(String presignedUrlString) {
    ByteArrayOutputStream byteArrayOutputStream = new
ByteArrayOutputStream(); // Capture the response body to a byte array.

    HttpRequest.Builder requestBuilder = HttpRequest.newBuilder();
    HttpClient httpClient = HttpClient.newHttpClient();
    try {
        URL presignedUrl = new URL(presignedUrlString);
        HttpResponse<InputStream> response = httpClient.send(requestBuilder
            .uri(presignedUrl.toURI())
            .GET()
            .build(),
            HttpResponse.BodyHandlers.ofInputStream());

        IoUtils.copy(response.body(), byteArrayOutputStream);

        logger.info("HTTP response code is " + response.statusCode());

    } catch (URISyntaxException | InterruptedException | IOException e) {
        logger.error(e.getMessage(), e);
    }
    return byteArrayOutputStream.toByteArray();
}

```

Utilizzate la classe AWS SDK for `SdkHttpClient` Java per eseguire il download.

```

/* Use the AWS SDK for Java SdkHttpClient class to do the download. */
public byte[] useSdkHttpClientToPut(String presignedUrlString) {

    ByteArrayOutputStream byteArrayOutputStream = new
ByteArrayOutputStream(); // Capture the response body to a byte array.
    try {
        URL presignedUrl = new URL(presignedUrlString);
        SdkHttpRequest request = SdkHttpRequest.builder()
            .method(SdkHttpMethod.GET)
            .uri(presignedUrl.toURI())
            .build();

        HttpExecuteRequest executeRequest = HttpExecuteRequest.builder()
            .request(request)
            .build();

        try (SdkHttpClient sdkHttpClient = ApacheHttpClient.create()) {
            HttpExecuteResponse response =
sdkHttpClient.prepareRequest(executeRequest).call();
            response.responseBody().ifPresentOrElse(
                abortableInputStream -> {
                    try {
                        IoUtils.copy(abortableInputStream,
byteArrayOutputStream);
                    } catch (IOException e) {
                        throw new RuntimeException(e);
                    }
                },
                () -> logger.error("No response body."));

            logger.info("HTTP Response code is {}",
response.httpResponse().statusCode());
        } catch (URISyntaxException | IOException e) {
            logger.error(e.getMessage(), e);
        }
        return byteArrayOutputStream.toByteArray();
    }
}

```

Genera un URL prefirmato per un caricamento, quindi carica un file (richiesta PUT).

Importazioni.

```
import com.example.s3.util.PresignUrlUtils;
import org.slf4j.Logger;
import software.amazon.awssdk.core.internal.sync.FileContentStreamProvider;
import software.amazon.awssdk.http.HttpExecuteRequest;
import software.amazon.awssdk.http.HttpExecuteResponse;
import software.amazon.awssdk.http.SdkHttpClient;
import software.amazon.awssdk.http.SdkHttpMethod;
import software.amazon.awssdk.http.SdkHttpRequest;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.presigner.S3Presigner;
import software.amazon.awssdk.services.s3.presigner.model.PresignedPutObjectRequest;
import software.amazon.awssdk.services.s3.presigner.model.PutObjectPresignRequest;

import java.io.File;
import java.io.IOException;
import java.io.OutputStream;
import java.io.RandomAccessFile;
import java.net.HttpURLConnection;
import java.net.URISyntaxException;
import java.net.URL;
import java.net.http.HttpClient;
import java.net.http.HttpRequest;
import java.net.http.HttpResponse;
import java.nio.ByteBuffer;
import java.nio.channels.FileChannel;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.time.Duration;
import java.util.Map;
import java.util.UUID;
```

## Genera l'URL.

```
/* Create a presigned URL to use in a subsequent PUT request */
public String createPresignedUrl(String bucketName, String keyName, Map<String,
String> metadata) {
    try (S3Presigner presigner = S3Presigner.create()) {

        PutObjectRequest objectRequest = PutObjectRequest.builder()
```

```

        .bucket(bucketName)
        .key(keyName)
        .metadata(metadata)
        .build();

    PutObjectPresignRequest presignRequest =
PutObjectPresignRequest.builder()
        .signatureDuration(Duration.ofMinutes(10)) // The URL expires
in 10 minutes.
        .putObjectRequest(objectRequest)
        .build();

    PresignedPutObjectRequest presignedRequest =
presigner.presignPutObject(presignRequest);
    String myURL = presignedRequest.url().toString();
    logger.info("Presigned URL to upload a file to: [{}]", myURL);
    logger.info("HTTP method: [{}]",
presignedRequest.httpRequest().method());

    return presignedRequest.url().toExternalForm();
}
}

```

Caricate un oggetto file utilizzando uno dei tre approcci seguenti.

Utilizzate la classe JDK `URLConnection` (dalla v1.1) per eseguire il caricamento.

```

/* Use the JDK HttpURLConnection (since v1.1) class to do the upload. */
public void useURLConnectionToPut(String presignedUrlString, File fileToPut,
Map<String, String> metadata) {
    logger.info("Begin [{}] upload", fileToPut.toString());
    try {
        URL presignedUrl = new URL(presignedUrlString);
        HttpURLConnection connection = (HttpURLConnection)
presignedUrl.openConnection();
        connection.setDoOutput(true);
        metadata.forEach((k, v) -> connection.setRequestProperty("x-amz-meta-" +
k, v));
        connection.setRequestMethod("PUT");
        OutputStream out = connection.getOutputStream();

        try (RandomAccessFile file = new RandomAccessFile(fileToPut, "r");

```



```

        FileChannel inChannel = file.getChannel() {
        ByteBuffer buffer = ByteBuffer.allocate(8192); //Buffer size is 8k

        while (inChannel.read(buffer) > 0) {
            buffer.flip();
            for (int i = 0; i < buffer.limit(); i++) {
                out.write(buffer.get());
            }
            buffer.clear();
        }
    } catch (IOException e) {
        logger.error(e.getMessage(), e);
    }

    out.close();
    connection.getResponseCode();
    logger.info("HTTP response code is " + connection.getResponseCode());

} catch (S3Exception | IOException e) {
    logger.error(e.getMessage(), e);
}
}

```

Utilizzate la classe JDK `HttpClient` (dalla v11) per eseguire il caricamento.

```

/* Use the JDK HttpClient (since v11) class to do the upload. */
public void useHttpClientToPut(String presignedUrlString, File fileToPut,
Map<String, String> metadata) {
    logger.info("Begin [{}] upload", fileToPut.toString());

    HttpRequest.Builder requestBuilder = HttpRequest.newBuilder();
    metadata.forEach((k, v) -> requestBuilder.header("x-amz-meta-" + k, v));

    HttpClient httpClient = HttpClient.newHttpClient();
    try {
        final HttpResponse<Void> response = httpClient.send(requestBuilder
            .uri(new URL(presignedUrlString).toURI())

        .PUT(HttpRequest.BodyPublishers.ofFile(Path.of(fileToPut.toURI()))
            .build(),
            HttpResponse.BodyHandlers.discarding());
    }
}

```

```

        logger.info("HTTP response code is " + response.statusCode());
    } catch (URISyntaxException | InterruptedException | IOException e) {
        logger.error(e.getMessage(), e);
    }
}

```

Utilizzate la `SdkHttpClient` classe AWS for Java V2 per eseguire il caricamento.

```

/* Use the AWS SDK for Java V2 SdkHttpClient class to do the upload. */
public void useSdkHttpClientToPut(String presignedUrlString, File fileToPut,
Map<String, String> metadata) {
    logger.info("Begin [{}] upload", fileToPut.toString());

    try {
        URL presignedUrl = new URL(presignedUrlString);

        SdkHttpRequest.Builder requestBuilder = SdkHttpRequest.builder()
            .method(SdkHttpMethod.PUT)
            .uri(presignedUrl.toURI());
        // Add headers
        metadata.forEach((k, v) -> requestBuilder.putHeader("x-amz-meta-" + k,
v));
        // Finish building the request.
        SdkHttpRequest request = requestBuilder.build();

        HttpExecuteRequest executeRequest = HttpExecuteRequest.builder()
            .request(request)
            .contentStreamProvider(new
FileContentStreamProvider(fileToPut.toPath()))
            .build();

        try (SdkHttpClient sdkHttpClient = ApacheHttpClient.create()) {
            HttpExecuteResponse response =
sdkHttpClient.prepareRequest(executeRequest).call();
            logger.info("Response code: {}",
response.httpResponse().statusCode());
        }
    } catch (URISyntaxException | IOException e) {
        logger.error(e.getMessage(), e);
    }
}

```

## Eliminare i caricamenti multiparte incompleti

Il seguente esempio di codice mostra come eliminare o interrompere i caricamenti multiparte incompleti di Amazon S3.

SDK per Java 2.x

### Note

C'è altro su [GitHub](#) Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Per interrompere i caricamenti in più parti che sono in corso o incompleti per qualsiasi motivo, puoi creare un elenco di caricamenti e quindi eliminarli come mostrato nell'esempio seguente.

```
public static void abortIncompleteMultipartUploadsFromList() {
    ListMultipartUploadsRequest listMultipartUploadsRequest =
ListMultipartUploadsRequest.builder()
        .bucket(bucketName)
        .build();

    ListMultipartUploadsResponse response =
s3Client.listMultipartUploads(listMultipartUploadsRequest);
    List<MultipartUpload> uploads = response.uploads();

    AbortMultipartUploadRequest abortMultipartUploadRequest;
    for (MultipartUpload upload : uploads) {
        abortMultipartUploadRequest = AbortMultipartUploadRequest.builder()
            .bucket(bucketName)
            .key(upload.key())
            .expectedBucketOwner(accountId)
            .uploadId(upload.uploadId())
            .build();

        AbortMultipartUploadResponse abortMultipartUploadResponse =
s3Client.abortMultipartUpload(abortMultipartUploadRequest);
        if (abortMultipartUploadResponse.sdkHttpResponse().isSuccessful()) {
```

```

        logger.info("Upload ID [{}] to bucket [{}] successfully aborted.",
upload.uploadId(), bucketName);
    }
}

```

Per eliminare i caricamenti incompleti in più parti iniziati prima o dopo una data, puoi eliminare selettivamente i caricamenti in più parti in base a un momento temporale, come mostrato nell'esempio seguente.

```

static void abortIncompleteMultipartUploadsOlderThan(Instant pointInTime) {
    ListMultipartUploadsRequest listMultipartUploadsRequest =
ListMultipartUploadsRequest.builder()
    .bucket(bucketName)
    .build();

    ListMultipartUploadsResponse response =
s3Client.listMultipartUploads(listMultipartUploadsRequest);
    List<MultipartUpload> uploads = response.uploads();

    AbortMultipartUploadRequest abortMultipartUploadRequest;
    for (MultipartUpload upload : uploads) {
        logger.info("Found multipartUpload with upload ID [{}], initiated [{}]",
upload.uploadId(), upload.initiated());
        if (upload.initiated().isBefore(pointInTime)) {
            abortMultipartUploadRequest = AbortMultipartUploadRequest.builder()
                .bucket(bucketName)
                .key(upload.key())
                .expectedBucketOwner(accountId)
                .uploadId(upload.uploadId())
                .build();

            AbortMultipartUploadResponse abortMultipartUploadResponse =
s3Client.abortMultipartUpload(abortMultipartUploadRequest);
            if (abortMultipartUploadResponse.sdkHttpResponse().isSuccessful()) {
                logger.info("Upload ID [{}] to bucket [{}] successfully
aborted.", upload.uploadId(), bucketName);
            }
        }
    }
}

```

Se hai accesso all'ID di caricamento dopo aver iniziato un caricamento in più parti, puoi eliminare il caricamento in corso utilizzando l'ID.

```
static void abortMultipartUploadUsingUploadId() {
    String uploadId = startUploadReturningUploadId();
    AbortMultipartUploadResponse response = s3Client.abortMultipartUpload(b -> b
        .uploadId(uploadId)
        .bucket(bucketName)
        .key(key));

    if (response.sdkHttpResponse().isSuccessful()) {
        logger.info("Upload ID [{}] to bucket [{}] successfully aborted.",
            uploadId, bucketName);
    }
}
```

Per eliminare in modo coerente i caricamenti incompleti in più parti più vecchi di un certo numero di giorni, imposta una configurazione del ciclo di vita del bucket per il bucket. L'esempio seguente mostra come creare una regola per eliminare i caricamenti incompleti più vecchi di 7 giorni.

```
static void abortMultipartUploadsUsingLifecycleConfig() {
    Collection<LifecycleRule> lifeCycleRules = List.of(LifecycleRule.builder()
        .abortIncompleteMultipartUpload(b -> b
            .daysAfterInitiation(7))
        .status("Enabled")
        .filter(SdkBuilder::build) // Filter element is required.
        .build());

    // If the action is successful, the service sends back an HTTP 200 response
    with an empty HTTP body.
    PutBucketLifecycleConfigurationResponse response =
    s3Client.putBucketLifecycleConfiguration(b -> b
        .bucket(bucketName)
        .lifecycleConfiguration(b1 -> b1.rules(lifeCycleRules)));

    if (response.sdkHttpResponse().isSuccessful()) {
        logger.info("Rule to abort incomplete multipart uploads added to
            bucket.");
    } else {
```

```
        logger.error("Unsuccessfully applied rule. HTTP status code is [{}]",
            response.sdkHttpResponse().statusCode());
    }
}
```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella Documentazione di riferimento delle API AWS SDK for Java 2.x .
  - [AbortMultipartUpload](#)
  - [ListMultipartUploads](#)
  - [PutBucketLifecycleConfiguration](#)

Scarica oggetti in una directory locale

L'esempio di codice seguente mostra come scaricare tutti gli oggetti da un bucket Amazon Simple Storage Service (Amazon S3) in una directory locale.

SDK per Java 2.x

#### Note

C'è altro da fare. [GitHub](#) Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Usa un [S3 TransferManager](#) per [scaricare tutti gli oggetti S3](#) nello stesso bucket S3. Visualizza il [file completo](#) ed esegui il [test](#).

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.services.s3.model.ObjectIdentifier;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.CompletedDirectoryDownload;
import software.amazon.awssdk.transfer.s3.model.DirectoryDownload;
import software.amazon.awssdk.transfer.s3.model.DownloadDirectoryRequest;
import java.io.IOException;
import java.net.URI;
import java.net.URISyntaxException;
import java.nio.file.Files;
```

```
import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.HashSet;
import java.util.Set;
import java.util.UUID;
import java.util.stream.Collectors;

public Integer downloadObjectsToDirectory(S3TransferManager transferManager,
    URI destinationPathURI, String bucketName) {
    DirectoryDownload directoryDownload =
transferManager.downloadDirectory(DownloadDirectoryRequest.builder()
    .destination(Paths.get(destinationPathURI))
    .bucket(bucketName)
    .build());
    CompletedDirectoryDownload completedDirectoryDownload =
directoryDownload.completionFuture().join();

    completedDirectoryDownload.failedTransfers()
        .forEach(fail -> logger.warn("Object [{}] failed to transfer",
fail.toString()));
    return completedDirectoryDownload.failedTransfers().size();
}
```

- Per i dettagli sull'API, consulta [DownloadDirectory](#) la sezione API Reference.AWS SDK for Java 2.x

## Nozioni di base su bucket e oggetti

L'esempio di codice seguente mostra come:

- Creare un bucket e caricare un file in tale bucket.
- Scaricare un oggetto da un bucket.
- Copiare un oggetto in una sottocartella in un bucket.
- Elencare gli oggetti in un bucket.
- Eliminare il bucket e tutti gli oggetti in esso contenuti.

## SDK per Java 2.x

 Note

C'è di più su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * This Java code example performs the following tasks:
 *
 * 1. Creates an Amazon S3 bucket.
 * 2. Uploads an object to the bucket.
 * 3. Downloads the object to another local file.
 * 4. Uploads an object using multipart upload.
 * 5. List all objects located in the Amazon S3 bucket.
 * 6. Copies the object to another Amazon S3 bucket.
 * 7. Deletes the object from the Amazon S3 bucket.
 * 8. Deletes the Amazon S3 bucket.
 */

public class S3Scenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    public static void main(String[] args) throws IOException {
        final String usage = ""

            Usage:
                <bucketName> <key> <objectPath> <savePath> <toBucket>

            Where:
                bucketName - The Amazon S3 bucket to create.
                key - The key to use.
                objectPath - The path where the file is located (for example,
                C:/AWS/book2.pdf).
    }
}
```



```
        savePath - The path where the file is saved after it's
downloaded (for example, C:/AWS/book2.pdf).
        toBucket - An Amazon S3 bucket to where an object is copied to
(for example, C:/AWS/book2.pdf).\s
        """;

    if (args.length != 5) {
        System.out.println(usage);
        System.exit(1);
    }

    String bucketName = args[0];
    String key = args[1];
    String objectPath = args[2];
    String savePath = args[3];
    String toBucket = args[4];
    Region region = Region.US_EAST_1;
    S3Client s3 = S3Client.builder()
        .region(region)
        .build();

    System.out.println(DASHES);
    System.out.println("Welcome to the Amazon S3 example scenario.");
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("1. Create an Amazon S3 bucket.");
    createBucket(s3, bucketName);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("2. Upload a local file to the Amazon S3 bucket.");
    uploadLocalFile(s3, bucketName, key, objectPath);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("3. Download the object to another local file.");
    getObjectBytes(s3, bucketName, key, savePath);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("4. Perform a multipart upload.");
    String multipartKey = "multiPartKey";
    multipartUpload(s3, toBucket, multipartKey);
```

```
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("5. List all objects located in the Amazon S3 bucket.");
listAllObjects(s3, bucketName);
anotherListExample(s3, bucketName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("6. Copy the object to another Amazon S3 bucket.");
copyBucketObject(s3, bucketName, key, toBucket);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("7. Delete the object from the Amazon S3 bucket.");
deleteObjectFromBucket(s3, bucketName, key);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("8. Delete the Amazon S3 bucket.");
deleteBucket(s3, bucketName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("All Amazon S3 operations were successfully performed");
System.out.println(DASHES);
s3.close();
}

// Create a bucket by using a S3Waiter object.
public static void createBucket(S3Client s3Client, String bucketName) {
    try {
        S3Waiter s3Waiter = s3Client.waiter();
        CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
            .bucket(bucketName)
            .build();

        s3Client.createBucket(bucketRequest);
        HeadBucketRequest bucketRequestWait = HeadBucketRequest.builder()
            .bucket(bucketName)
            .build();

        // Wait until the bucket is created and print out the response.
```

```
        WaiterResponse<HeadBucketResponse> waiterResponse =
s3Waiter.waitUntilBucketExists(bucketRequestWait);
        waiterResponse.matched().response().ifPresent(System.out::println);
        System.out.println(bucketName + " is ready");

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void deleteBucket(S3Client client, String bucket) {
    DeleteBucketRequest deleteBucketRequest = DeleteBucketRequest.builder()
        .bucket(bucket)
        .build();

    client.deleteBucket(deleteBucketRequest);
    System.out.println(bucket + " was deleted.");
}

/**
 * Upload an object in parts.
 */
public static void multipartUpload(S3Client s3, String bucketName, String key) {
    int mB = 1024 * 1024;
    // First create a multipart upload and get the upload id.
    CreateMultipartUploadRequest createMultipartUploadRequest =
CreateMultipartUploadRequest.builder()
        .bucket(bucketName)
        .key(key)
        .build();

    CreateMultipartUploadResponse response =
s3.createMultipartUpload(createMultipartUploadRequest);
    String uploadId = response.uploadId();
    System.out.println(uploadId);

    // Upload all the different parts of the object.
    UploadPartRequest uploadPartRequest1 = UploadPartRequest.builder()
        .bucket(bucketName)
        .key(key)
        .uploadId(uploadId)
        .partNumber(1).build();
```

```
String etag1 = s3.uploadPart(uploadPartRequest1,
    RequestBody.fromByteBuffer(getRandomByteBuffer(5 * MB)))
    .eTag();
CompletedPart part1 =
CompletedPart.builder().partNumber(1).eTag(etag1).build();

UploadPartRequest uploadPartRequest2 =
UploadPartRequest.builder().bucket(bucketName).key(key)
    .uploadId(uploadId)
    .partNumber(2).build();
String etag2 = s3.uploadPart(uploadPartRequest2,
    RequestBody.fromByteBuffer(getRandomByteBuffer(3 * MB)))
    .eTag();
CompletedPart part2 =
CompletedPart.builder().partNumber(2).eTag(etag2).build();

// Call completeMultipartUpload operation to tell S3 to merge all uploaded
// parts and finish the multipart operation.
CompletedMultipartUpload completedMultipartUpload =
CompletedMultipartUpload.builder()
    .parts(part1, part2)
    .build();

CompleteMultipartUploadRequest completeMultipartUploadRequest =
CompleteMultipartUploadRequest.builder()
    .bucket(bucketName)
    .key(key)
    .uploadId(uploadId)
    .multipartUpload(completedMultipartUpload)
    .build();

s3.completeMultipartUpload(completeMultipartUploadRequest);
}

private static ByteBuffer getRandomByteBuffer(int size) {
    byte[] b = new byte[size];
    new Random().nextBytes(b);
    return ByteBuffer.wrap(b);
}

public static void getObjectBytes(S3Client s3, String bucketName, String
keyName, String path) {
    try {
        GetObjectRequest objectRequest = GetObjectRequest
```

```
        .builder()
        .key(keyName)
        .bucket(bucketName)
        .build();

    ResponseBytes<GetObjectResponse> objectBytes =
s3.getObjectAsBytes(objectRequest);
    byte[] data = objectBytes.asByteArray();

    // Write the data to a local file.
    File myFile = new File(path);
    OutputStream os = new FileOutputStream(myFile);
    os.write(data);
    System.out.println("Successfully obtained bytes from an S3 object");
    os.close();

    } catch (IOException ex) {
        ex.printStackTrace();
    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void uploadLocalFile(S3Client s3, String bucketName, String key,
String objectPath) {
    PutObjectRequest objectRequest = PutObjectRequest.builder()
        .bucket(bucketName)
        .key(key)
        .build();

    s3.putObject(objectRequest, RequestBody.fromFile(new File(objectPath)));
}

public static void listAllObjects(S3Client s3, String bucketName) {
    ListObjectsV2Request listObjectsReqManual = ListObjectsV2Request.builder()
        .bucket(bucketName)
        .maxKeys(1)
        .build();

    boolean done = false;
    while (!done) {
        ListObjectsV2Response listObjResponse =
s3.listObjectsV2(listObjectsReqManual);
```

```
        for (S3Object content : listObjResponse.contents()) {
            System.out.println(content.key());
        }

        if (listObjResponse.nextContinuationToken() == null) {
            done = true;
        }

        listObjectsReqManual = listObjectsReqManual.toBuilder()
            .continuationToken(listObjResponse.nextContinuationToken())
            .build();
    }
}

public static void anotherListExample(S3Client s3, String bucketName) {
    ListObjectsV2Request listReq = ListObjectsV2Request.builder()
        .bucket(bucketName)
        .maxKeys(1)
        .build();

    ListObjectsV2Iterable listRes = s3.listObjectsV2Paginator(listReq);

    // Process response pages.
    listRes.stream()
        .flatMap(r -> r.contents().stream())
        .forEach(content -> System.out.println(" Key: " + content.key() + "
size = " + content.size()));

    // Helper method to work with paginated collection of items directly.
    listRes.contents().stream()
        .forEach(content -> System.out.println(" Key: " + content.key() + "
size = " + content.size()));

    for (S3Object content : listRes.contents()) {
        System.out.println(" Key: " + content.key() + " size = " +
content.size());
    }
}

public static void deleteObjectFromBucket(S3Client s3, String bucketName, String
key) {
    DeleteObjectRequest deleteObjectRequest = DeleteObjectRequest.builder()
        .bucket(bucketName)
        .key(key)
```

```
        .build();

        s3.deleteObject(deleteObjectRequest);
        System.out.println(key + " was deleted");
    }

    public static String copyBucketObject(S3Client s3, String fromBucket, String
objectKey, String toBucket) {
        String encodedUrl = null;
        try {
            encodedUrl = URLEncoder.encode(fromBucket + "/" + objectKey,
StandardCharsets.UTF_8.toString());
        } catch (UnsupportedEncodingException e) {
            System.out.println("URL could not be encoded: " + e.getMessage());
        }
        CopyObjectRequest copyReq = CopyObjectRequest.builder()
            .copySource(encodedUrl)
            .destinationBucket(toBucket)
            .destinationKey(objectKey)
            .build();

        try {
            CopyObjectResponse copyRes = s3.copyObject(copyReq);
            System.out.println("The " + objectKey + " was copied to " + toBucket);
            return copyRes.copyObjectResult().toString();

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
}
```


- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella Documentazione di riferimento delle API AWS SDK for Java 2.x .
  - [CopyObject](#)
  - [CreateBucket](#)
  - [DeleteBucket](#)
  - [DeleteObjects](#)

- [GetObject](#)
- [ListObjectsV2](#)
- [PutObject](#)

Ottieni la configurazione di conservazione legale di un oggetto

Il seguente esempio di codice mostra come ottenere la configurazione di conservazione legale di un bucket S3.

SDK per Java 2.x

 Note

C'è di più su [GitHub](#) Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
// Get the legal hold details for an S3 object.
public ObjectLockLegalHold getObjectLegalHold(String bucketName, String
objectKey) {
    try {
        GetObjectLegalHoldRequest legalHoldRequest =
GetObjectLegalHoldRequest.builder()
            .bucket(bucketName)
            .key(objectKey)
            .build();

        GetObjectLegalHoldResponse response =
getClient().getObjectLegalHold(legalHoldRequest);
        System.out.println("Object legal hold for " + objectKey + " in " +
bucketName +
            ":\n\tStatus: " + response.legalHold().status());
        return response.legalHold();

    } catch (S3Exception ex) {
        System.out.println("\tUnable to fetch legal hold: '" + ex.getMessage() +
        "'");
    }

    return null;
}
```



```
}
```

- Per i dettagli sull'API, consulta la [GetObjectLegalHold](#) sezione AWS SDK for Java 2.x API Reference.

## Blocca oggetti Amazon S3

Il seguente esempio di codice mostra come utilizzare le funzionalità di blocco degli oggetti di S3.

### SDK per Java 2.x

#### Note

C'è altro su GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Esegui uno scenario interattivo che dimostri le funzionalità di blocco degli oggetti di Amazon S3.

```
import software.amazon.awssdk.services.s3.model.ObjectLockLegalHold;
import software.amazon.awssdk.services.s3.model.ObjectLockRetention;
import java.io.BufferedWriter;
import java.io.IOException;
import java.time.LocalDateTime;
import java.time.format.DateTimeFormatter;
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;
import java.util.stream.Collectors;
```

```
/*
```

Before running this Java V2 code example, set up your development environment, including your credentials.

For more information, see the following documentation topic:

<https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/setup.html>

This Java example performs the following tasks:

1. Create test Amazon Simple Storage Service (S3) buckets with different lock policies.
2. Upload sample objects to each bucket.

3. Set some Legal Hold and Retention Periods on objects and buckets.
4. Investigate lock policies by viewing settings or attempting to delete or overwrite objects.
5. Clean up objects and buckets.

\*/

```
public class S3ObjectLockWorkflow {

    public static final String DASHES = new String(new char[80]).replace("\0", "-");
    static String bucketName;
    static S3LockActions s3LockActions;
    private static final List<String> bucketNames = new ArrayList<>();
    private static final List<String> fileNames = new ArrayList<>();

    public static void main(String[] args) {
        // Get the current date and time to ensure bucket name is unique.
        LocalDateTime currentTime = LocalDateTime.now();

        // Format the date and time as a string.
        DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyyMMddHHmmss");
        String timeStamp = currentTime.format(formatter);

        s3LockActions = new S3LockActions();
        bucketName = "bucket"+timeStamp;
        Scanner scanner = new Scanner(System.in);

        System.out.println(DASHES);
        System.out.println("Welcome to the Amazon Simple Storage Service (S3) Object
Locking Workflow Scenario.");
        System.out.println("Press Enter to continue...");
        scanner.nextLine();
        configurationSetup();
        System.out.println(DASHES);

        System.out.println(DASHES);
        setup();
        System.out.println("Setup is complete. Press Enter to continue...");
        scanner.nextLine();
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Lets present the user with choices.");
        System.out.println("Press Enter to continue...");
        scanner.nextLine();
        demoActionChoices() ;
    }
}
```

```
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Would you like to clean up the resources? (y/n)");
        String delAns = scanner.nextLine().trim();
        if (delAns.equalsIgnoreCase("y")) {
            cleanup();
            System.out.println("Clean up is complete.");
        }

        System.out.println("Press Enter to continue...");
        scanner.nextLine();
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Amazon S3 Object Locking Workflow is complete.");
        System.out.println(DASHES);
    }

    // Present the user with the demo action choices.
    public static void demoActionChoices() {
        String[] choices = {
            "List all files in buckets.",
            "Attempt to delete a file.",
            "Attempt to delete a file with retention period bypass.",
            "Attempt to overwrite a file.",
            "View the object and bucket retention settings for a file.",
            "View the legal hold settings for a file.",
            "Finish the workflow."
        };

        int choice = 0;
        while (true) {
            System.out.println(DASHES);
            choice = getChoiceResponse("Explore the S3 locking features by selecting
one of the following choices:", choices);
            System.out.println(DASHES);
            System.out.println("You selected "+choices[choice]);
            switch (choice) {
                case 0 -> {
                    s3LockActions.listBucketsAndObjects(bucketNames, true);
                }

                case 1 -> {
```

```
        System.out.println("Enter the number of the object to delete:");
        List<S3InfoObject> allFiles =
s3LockActions.listBucketsAndObjects(bucketNames, true);
        List<String> fileKeys = allFiles.stream().map(f ->
f.getKeyName()).collect(Collectors.toList());
        String[] fileKeysArray = fileKeys.toArray(new String[0]);
        int fileChoice = getChoiceResponse(null, fileKeysArray);
        String objectKey = fileKeys.get(fileChoice);
        String bucketName = allFiles.get(fileChoice).getBucketName();
        String version = allFiles.get(fileChoice).getVersion();
        s3LockActions.deleteObjectFromBucket(bucketName, objectKey,
false, version);
    }

    case 2 -> {
        System.out.println("Enter the number of the object to delete:");
        List<S3InfoObject> allFiles =
s3LockActions.listBucketsAndObjects(bucketNames, true);
        List<String> fileKeys = allFiles.stream().map(f ->
f.getKeyName()).collect(Collectors.toList());
        String[] fileKeysArray = fileKeys.toArray(new String[0]);
        int fileChoice = getChoiceResponse(null, fileKeysArray);
        String objectKey = fileKeys.get(fileChoice);
        String bucketName = allFiles.get(fileChoice).getBucketName();
        String version = allFiles.get(fileChoice).getVersion();
        s3LockActions.deleteObjectFromBucket(bucketName, objectKey,
true, version);
    }

    case 3 -> {
        System.out.println("Enter the number of the object to
overwrite:");
        List<S3InfoObject> allFiles =
s3LockActions.listBucketsAndObjects(bucketNames, true);
        List<String> fileKeys = allFiles.stream().map(f ->
f.getKeyName()).collect(Collectors.toList());
        String[] fileKeysArray = fileKeys.toArray(new String[0]);
        int fileChoice = getChoiceResponse(null, fileKeysArray);
        String objectKey = fileKeys.get(fileChoice);
        String bucketName = allFiles.get(fileChoice).getBucketName();

        // Attempt to overwrite the file.
        try (BufferedWriter writer = new BufferedWriter(new
java.io.FileWriter(objectKey))) {
```

```
        writer.write("This is a modified text.");

    } catch (IOException e) {
        e.printStackTrace();
    }
    s3LockActions.uploadFile(bucketName, objectKey, objectKey);
}

case 4 -> {
    System.out.println("Enter the number of the object to
overwrite:");
    List<S3InfoObject> allFiles =
s3LockActions.listBucketsAndObjects(bucketNames, true);
    List<String> fileKeys = allFiles.stream().map(f ->
f.getKeyName()).collect(Collectors.toList());
    String[] fileKeysArray = fileKeys.toArray(new String[0]);
    int fileChoice = getChoiceResponse(null, fileKeysArray);
    String objectKey = fileKeys.get(fileChoice);
    String bucketName = allFiles.get(fileChoice).getBucketName();
    s3LockActions.getObjectRetention(bucketName, objectKey);
}

case 5 -> {
    System.out.println("Enter the number of the object to view:");
    List<S3InfoObject> allFiles =
s3LockActions.listBucketsAndObjects(bucketNames, true);
    List<String> fileKeys = allFiles.stream().map(f ->
f.getKeyName()).collect(Collectors.toList());
    String[] fileKeysArray = fileKeys.toArray(new String[0]);
    int fileChoice = getChoiceResponse(null, fileKeysArray);
    String objectKey = fileKeys.get(fileChoice);
    String bucketName = allFiles.get(fileChoice).getBucketName();
    s3LockActions.getObjectLegalHold(bucketName, objectKey);
    s3LockActions.getBucketObjectLockConfiguration(bucketName);
}

case 6 -> {
    System.out.println("Exiting the workflow...");
    return;
}

default -> {
    System.out.println("Invalid choice. Please select again.");
}
}
```

```

    }
  }
}

// Clean up the resources from the scenario.
private static void cleanup() {
    List<S3InfoObject> allFiles =
s3LockActions.listBucketsAndObjects(bucketNames, false);
    for (S3InfoObject fileInfo : allFiles) {
        String bucketName = fileInfo.getBucketName();
        String key = fileInfo.getKeyName();
        String version = fileInfo.getVersion();
        if (bucketName.contains("lock-enabled") ||
(bucketName.contains("retention-after-creation"))) {
            ObjectLockLegalHold legalHold =
s3LockActions.getObjectLegalHold(bucketName, key);
            if (legalHold != null) {
                String holdStatus = legalHold.status().name();
                System.out.println(holdStatus);
                if (holdStatus.compareTo("ON") == 0) {
                    s3LockActions.modifyObjectLegalHold(bucketName, key, false);
                }
            }
            // Check for a retention period.
            ObjectLockRetention retention =
s3LockActions.getObjectRetention(bucketName, key);
            boolean hasRetentionPeriod ;
            hasRetentionPeriod = retention != null;
            s3LockActions.deleteObjectFromBucket(bucketName,
key,hasRetentionPeriod, version);

        } else {
            System.out.println(bucketName + " objects do not have a legal lock");
            s3LockActions.deleteObjectFromBucket(bucketName, key,false,
version);
        }
    }
}

// Delete the buckets.
System.out.println("Delete "+bucketName);
for (String bucket : bucketNames){
    s3LockActions.deleteBucketByName(bucket);
}
}

```

```
private static void setup() {
    Scanner scanner = new Scanner(System.in);
    System.out.println("""
        For this workflow, we will use the AWS SDK for Java to create
several S3
        buckets and files to demonstrate working with S3 locking features.
        """);

    System.out.println("S3 buckets can be created either with or without object
lock enabled.");
    System.out.println("Press Enter to continue...");
    scanner.nextLine();

    // Create three S3 buckets.
    s3LockActions.createBucketWithLockOptions(false, bucketNames.get(0));
    s3LockActions.createBucketWithLockOptions(true, bucketNames.get(1));
    s3LockActions.createBucketWithLockOptions(false, bucketNames.get(2));
    System.out.println("Press Enter to continue.");
    scanner.nextLine();

    System.out.println("Bucket "+bucketNames.get(2) +" will be configured to use
object locking with a default retention period.");
    s3LockActions.modifyBucketDefaultRetention(bucketNames.get(2));
    System.out.println("Press Enter to continue.");
    scanner.nextLine();

    System.out.println("Object lock policies can also be added to existing
buckets. For this example, we will use "+bucketNames.get(1));
    s3LockActions.enableObjectLockOnBucket(bucketNames.get(1));
    System.out.println("Press Enter to continue.");
    scanner.nextLine();

    // Upload some files to the buckets.
    System.out.println("Now let's add some test files:");
    String fileName = "exampleFile.txt";
    int fileCount = 2;
    try (BufferedWriter writer = new BufferedWriter(new
java.io.FileWriter(fileName))) {
        writer.write("This is a sample file for uploading to a bucket.");

    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

```
    for (String bucketName : bucketNames){
        for (int i = 0; i < fileCount; i++) {
            // Get the file name without extension.
            String fileNameWithoutExtension =
java.nio.file.Paths.get(fileName).getFileName().toString();
            int extensionIndex = fileNameWithoutExtension.lastIndexOf('.');
            if (extensionIndex > 0) {
                fileNameWithoutExtension = fileNameWithoutExtension.substring(0,
extensionIndex);
            }

            // Create the numbered file names.
            String numberedFileName = fileNameWithoutExtension + i +
getFileExtension(fileName);
            fileNames.add(numberedFileName);
            s3LockActions.uploadFile(bucketName, numberedFileName, fileName);
        }
    }

    String question = null;
    System.out.print("Press Enter to continue...");
    scanner.nextLine();
    System.out.println("Now we can set some object lock policies on individual
files:");
    for (String bucketName : bucketNames) {
        for (int i = 0; i < fileNames.size(); i++){

            // No modifications to the objects in the first bucket.
            if (!bucketName.equals(bucketNames.get(0))) {
                String exampleFileName = fileNames.get(i);
                switch (i) {
                    case 0 -> {
                        question = "Would you like to add a legal hold to " +
exampleFileName + " in " + bucketName + " (y/n)?";
                        System.out.println(question);
                        String ans = scanner.nextLine().trim();
                        if (ans.equalsIgnoreCase("y")) {
                            System.out.println("**** You have selected to put a
legal hold " + exampleFileName);

                                // Set a legal hold.
                                s3LockActions.modifyObjectLegalHold(bucketName,
exampleFileName, true);
```



```

        }
    }
    case 1 -> {
        ""
        Would you like to add a 1 day Governance retention
period to %s in %s (y/n)?
        Reminder: Only a user with the
s3:BypassGovernanceRetention permission will be able to delete this file or its
bucket until the retention period has expired.
        "".formatted(exampleFileName, bucketName);
        System.out.println(question);
        String ans2 = scanner.nextLine().trim();
        if (ans2.equalsIgnoreCase("y")) {

s3LockActions.modifyObjectRetentionPeriod(bucketName, exampleFileName);
        }
    }
}
}
}
}
}

// Get file extension.
private static String getFileExtension(String fileName) {
    int dotIndex = fileName.lastIndexOf('.');
    if (dotIndex > 0) {
        return fileName.substring(dotIndex);
    }
    return "";
}

public static void configurationSetup() {
    String noLockBucketName = bucketName + "-no-lock";
    String lockEnabledBucketName = bucketName + "-lock-enabled";
    String retentionAfterCreationBucketName = bucketName + "-retention-after-
creation";
    bucketNames.add(noLockBucketName);
    bucketNames.add(lockEnabledBucketName);
    bucketNames.add(retentionAfterCreationBucketName);
}

public static int getChoiceResponse(String question, String[] choices) {
    Scanner scanner = new Scanner(System.in);

```

```
        if (question != null) {
            System.out.println(question);
            for (int i = 0; i < choices.length; i++) {
                System.out.println("\t" + (i + 1) + ". " + choices[i]);
            }
        }

        int choiceNumber = 0;
        while (choiceNumber < 1 || choiceNumber > choices.length) {
            String choice = scanner.nextLine();
            try {
                choiceNumber = Integer.parseInt(choice);
            } catch (NumberFormatException e) {
                System.out.println("Invalid choice. Please enter a valid number.");
            }
        }

        return choiceNumber - 1;
    }
}
```

Una classe wrapper per le funzioni S3.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.BucketVersioningStatus;
import software.amazon.awssdk.services.s3.model.ChecksumAlgorithm;
import software.amazon.awssdk.services.s3.model.CreateBucketRequest;
import software.amazon.awssdk.services.s3.model.DefaultRetention;
import software.amazon.awssdk.services.s3.model.DeleteBucketRequest;
import software.amazon.awssdk.services.s3.model.DeleteObjectRequest;
import software.amazon.awssdk.services.s3.model.GetObjectLegalHoldRequest;
import software.amazon.awssdk.services.s3.model.GetObjectLegalHoldResponse;
import software.amazon.awssdk.services.s3.model.GetObjectLockConfigurationRequest;
import software.amazon.awssdk.services.s3.model.GetObjectLockConfigurationResponse;
import software.amazon.awssdk.services.s3.model.GetObjectRetentionRequest;
import software.amazon.awssdk.services.s3.model.GetObjectRetentionResponse;
import software.amazon.awssdk.services.s3.model.HeadBucketRequest;
import software.amazon.awssdk.services.s3.model.ListObjectVersionsRequest;
import software.amazon.awssdk.services.s3.model.ListObjectVersionsResponse;
import software.amazon.awssdk.services.s3.model.MFADelete;
import software.amazon.awssdk.services.s3.model.ObjectLockConfiguration;
```

```
import software.amazon.awssdk.services.s3.model.ObjectLockEnabled;
import software.amazon.awssdk.services.s3.model.ObjectLockLegalHold;
import software.amazon.awssdk.services.s3.model.ObjectLockLegalHoldStatus;
import software.amazon.awssdk.services.s3.model.ObjectLockRetention;
import software.amazon.awssdk.services.s3.model.ObjectLockRetentionMode;
import software.amazon.awssdk.services.s3.model.ObjectLockRule;
import software.amazon.awssdk.services.s3.model.PutBucketVersioningRequest;
import software.amazon.awssdk.services.s3.model.PutObjectLegalHoldRequest;
import software.amazon.awssdk.services.s3.model.PutObjectLockConfigurationRequest;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.model.PutObjectResponse;
import software.amazon.awssdk.services.s3.model.PutObjectRetentionRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.VersioningConfiguration;
import software.amazon.awssdk.services.s3.waiters.S3Waiter;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.time.Instant;
import java.time.ZoneId;
import java.time.ZonedDateTime;
import java.time.format.DateTimeFormatter;
import java.time.temporal.ChronoUnit;
import java.util.List;
import java.util.concurrent.atomic.AtomicInteger;
import java.util.stream.Collectors;

// Contains application logic for the Amazon S3 operations used in this workflow.
public class S3LockActions {

    private static S3Client getClient() {
        return S3Client.builder()
            .region(Region.US_EAST_1)
            .build();
    }

    // Set or modify a retention period on an object in an S3 bucket.
    public void modifyObjectRetentionPeriod(String bucketName, String objectKey) {
        // Calculate the instant one day from now.
        Instant futureInstant = Instant.now().plus(1, ChronoUnit.DAYS);

        // Convert the Instant to a ZonedDateTime object with a specific time zone.
        ZonedDateTime zonedDateTime = futureInstant.atZone(ZoneId.systemDefault());

        // Define a formatter for human-readable output.
```

```
        DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd
HH:mm:ss");

        // Format the ZonedDateTime object to a human-readable date string.
        String humanReadableDate = formatter.format(zonedDateTime);

        // Print the formatted date string.
        System.out.println("Formatted Date: " + humanReadableDate);
        ObjectLockRetention retention = ObjectLockRetention.builder()
            .mode(ObjectLockRetentionMode.GOVERNANCE)
            .retainUntilDate(futureInstant)
            .build();

        PutObjectRetentionRequest retentionRequest =
PutObjectRetentionRequest.builder()
            .bucket(bucketName)
            .key(objectKey)
            .retention(retention)
            .build();

        getClient().putObjectRetention(retentionRequest);
        System.out.println("Set retention for "+objectKey +" in " +bucketName +"
until "+ humanReadableDate +".");
    }

    // Get the legal hold details for an S3 object.
    public ObjectLockLegalHold getObjectLegalHold(String bucketName, String
objectKey) {
        try {
            GetObjectLegalHoldRequest legalHoldRequest =
GetObjectLegalHoldRequest.builder()
                .bucket(bucketName)
                .key(objectKey)
                .build();

            GetObjectLegalHoldResponse response =
getClient().getObjectLegalHold(legalHoldRequest);
            System.out.println("Object legal hold for " + objectKey + " in " +
bucketName +
                ":\n\tStatus: " + response.legalHold().status());
            return response.legalHold();

        } catch (S3Exception ex) {
```

```
        System.out.println("\tUnable to fetch legal hold: '" + ex.getMessage() +
        "");
    }

    return null;
}

// Create a new Amazon S3 bucket with object lock options.
public void createBucketWithLockOptions(boolean enableObjectLock, String
bucketName) {
    S3Waiter s3Waiter = getClient().waiter();
    CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
        .bucket(bucketName)
        .objectLockEnabledForBucket(enableObjectLock)
        .build();

    getClient().createBucket(bucketRequest);
    HeadBucketRequest bucketRequestWait = HeadBucketRequest.builder()
        .bucket(bucketName)
        .build();

    // Wait until the bucket is created and print out the response.
    s3Waiter.waitUntilBucketExists(bucketRequestWait);
    System.out.println(bucketName + " is ready");
}

public List<S3InfoObject> listBucketsAndObjects(List<String> bucketNames,
Boolean interactive) {
    AtomicInteger counter = new AtomicInteger(0); // Initialize counter.
    return bucketNames.stream()
        .flatMap(bucketName ->
listBucketObjectsAndVersions(bucketName).versions().stream()
        .map(version -> {
            S3InfoObject s3InfoObject = new S3InfoObject();
            s3InfoObject.setBucketName(bucketName);
            s3InfoObject.setVersion(version.versionId());
            s3InfoObject.setKeyName(version.key());
            return s3InfoObject;
        })))
        .peek(s3InfoObject -> {
            int i = counter.incrementAndGet(); // Increment and get the updated
value.

            if (interactive) {
                System.out.println(i + ": " + s3InfoObject.getKeyName());
            }
        });
}
```

```
        System.out.printf("%5s Bucket name: %s\n", "",
s3InfoObject.getBucketName());
        System.out.printf("%5s Version: %s\n", "",
s3InfoObject.getVersion());
    }
})
.collect(Collectors.toList());
}

public ListObjectVersionsResponse listBucketObjectsAndVersions(String
bucketName) {
    ListObjectVersionsRequest versionsRequest =
ListObjectVersionsRequest.builder()
        .bucket(bucketName)
        .build();

    return getClient().listObjectVersions(versionsRequest);
}

// Set or modify a retention period on an S3 bucket.
public void modifyBucketDefaultRetention(String bucketName) {
    VersioningConfiguration versioningConfiguration =
VersioningConfiguration.builder()
        .mfaDelete(MFADelete.DISABLED)
        .status(BucketVersioningStatus.ENABLED)
        .build();

    PutBucketVersioningRequest versioningRequest =
PutBucketVersioningRequest.builder()
        .bucket(bucketName)
        .versioningConfiguration(versioningConfiguration)
        .build();

    getClient().putBucketVersioning(versioningRequest);
    DefaultRetention retention = DefaultRetention.builder()
        .days(1)
        .mode(ObjectLockRetentionMode.GOVERNANCE)
        .build();

    ObjectLockRule lockRule = ObjectLockRule.builder()
        .defaultRetention(retention)
        .build();
}
```

```
        ObjectLockConfiguration objectLockConfiguration =
ObjectLockConfiguration.builder()
        .objectLockEnabled(ObjectLockEnabled.ENABLED)
        .rule(lockRule)
        .build();

        PutObjectLockConfigurationRequest putObjectLockConfigurationRequest =
PutObjectLockConfigurationRequest.builder()
        .bucket(bucketName)
        .objectLockConfiguration(objectLockConfiguration)
        .build();

        getClient().putObjectLockConfiguration(putObjectLockConfigurationRequest) ;
        System.out.println("Added a default retention to bucket "+bucketName +".");
    }

    // Enable object lock on an existing bucket.
    public void enableObjectLockOnBucket(String bucketName) {
        try {
            VersioningConfiguration versioningConfiguration =
VersioningConfiguration.builder()
                .status(BucketVersioningStatus.ENABLED)
                .build();

            PutBucketVersioningRequest putBucketVersioningRequest =
PutBucketVersioningRequest.builder()
                .bucket(bucketName)
                .versioningConfiguration(versioningConfiguration)
                .build();

            // Enable versioning on the bucket.
            getClient().putBucketVersioning(putBucketVersioningRequest);
            PutObjectLockConfigurationRequest request =
PutObjectLockConfigurationRequest.builder()
                .bucket(bucketName)
                .objectLockConfiguration(ObjectLockConfiguration.builder()
                    .objectLockEnabled(ObjectLockEnabled.ENABLED)
                    .build())
                .build();

            getClient().putObjectLockConfiguration(request);
            System.out.println("Successfully enabled object lock on "+bucketName);

        } catch (S3Exception ex) {
```

```
        System.out.println("Error modifying object lock: '" + ex.getMessage() +
        """);
    }
}

public void uploadFile(String bucketName, String objectName, String filePath) {
    Path file = Paths.get(filePath);
    PutObjectRequest request = PutObjectRequest.builder()
        .bucket(bucketName)
        .key(objectName)
        .checksumAlgorithm(ChecksumAlgorithm.SHA256)
        .build();

    PutObjectResponse response = getClient().putObject(request, file);
    if (response != null) {
        System.out.println("\tSuccessfully uploaded " + objectName + " to " +
        bucketName + ".");
    } else {
        System.out.println("\tCould not upload " + objectName + " to " +
        bucketName + ".");
    }
}

// Set or modify a legal hold on an object in an S3 bucket.
public void modifyObjectLegalHold(String bucketName, String objectKey, boolean
legalHoldOn) {
    ObjectLockLegalHold legalHold ;
    if (legalHoldOn) {
        legalHold = ObjectLockLegalHold.builder()
            .status(ObjectLockLegalHoldStatus.ON)
            .build();
    } else {
        legalHold = ObjectLockLegalHold.builder()
            .status(ObjectLockLegalHoldStatus.OFF)
            .build();
    }

    PutObjectLegalHoldRequest legalHoldRequest =
    PutObjectLegalHoldRequest.builder()
        .bucket(bucketName)
        .key(objectKey)
        .legalHold(legalHold)
        .build();
}
```



```
        getClient().putObjectLegalHold(legalHoldRequest) ;
        System.out.println("Modified legal hold for "+ objectKey +" in "+bucketName
+".");
    }

    // Delete an object from a specific bucket.
    public void deleteObjectFromBucket(String bucketName, String objectKey, boolean
hasRetention, String versionId) {
        try {
            DeleteObjectRequest objectRequest;
            if (hasRetention) {
                objectRequest = DeleteObjectRequest.builder()
                    .bucket(bucketName)
                    .key(objectKey)
                    .versionId(versionId)
                    .bypassGovernanceRetention(true)
                    .build();
            } else {
                objectRequest = DeleteObjectRequest.builder()
                    .bucket(bucketName)
                    .key(objectKey)
                    .versionId(versionId)
                    .build();
            }

            getClient().deleteObject(objectRequest) ;
            System.out.println("The object was successfully deleted");

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }
    }

    // Get the retention period for an S3 object.
    public ObjectLockRetention getObjectRetention(String bucketName, String key){
        try {
            GetObjectRetentionRequest retentionRequest =
GetObjectRetentionRequest.builder()
                .bucket(bucketName)
                .key(key)
                .build();

            GetObjectRetentionResponse response =
getClient().getObjectRetention(retentionRequest);
```

```
        System.out.println("\tObject retention for "+key+" in "+ bucketName +":
" + response.retention().mode() +" until "+ response.retention().retainUntilDate()
+ ".");
        return response.retention();

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        return null;
    }
}

public void deleteBucketByName(String bucketName) {
    try {
        DeleteBucketRequest request = DeleteBucketRequest.builder()
            .bucket(bucketName)
            .build();

        getClient().deleteBucket(request);
        System.out.println(bucketName +" was deleted.");

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }
}

// Get the object lock configuration details for an S3 bucket.
public void getBucketObjectLockConfiguration(String bucketName) {
    GetObjectLockConfigurationRequest objectLockConfigurationRequest =
GetObjectLockConfigurationRequest.builder()
        .bucket(bucketName)
        .build();

    GetObjectLockConfigurationResponse response =
getClient().getObjectLockConfiguration(objectLockConfigurationRequest);
    System.out.println("Bucket object lock config for "+bucketName +": ");
    System.out.println("\tEnabled:
"+response.getObjectLockConfiguration().getObjectLockEnabled());
    System.out.println("\tRule: "+
response.getObjectLockConfiguration().rule().defaultRetention());
}
}
```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella Documentazione di riferimento delle API AWS SDK for Java 2.x .
  - [GetObjectLegalHold](#)
  - [GetObjectLockConfiguration](#)
  - [GetObjectRetention](#)
  - [PutObjectLegalHold](#)
  - [PutObjectLockConfiguration](#)
  - [PutObjectRetention](#)

## Analizza gli URI

L'esempio di codice seguente mostra come analizzare gli URI di Amazon S3 per estrarre componenti importanti, come il nome del bucket e la chiave dell'oggetto.

## SDK per Java 2.x

### Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Analizza un URI Amazon S3 utilizzando la classe [S3Uri](#).

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.S3Uri;
import software.amazon.awssdk.services.s3.S3Utilities;

import java.net.URI;
import java.util.List;
import java.util.Map;

/**
 *
 * @param s3Client - An S3Client through which you acquire an S3Uri instance.
 * @param s3objectUrl - A complex URL (String) that is used to demonstrate S3Uri
```

```

    *                               capabilities.
    */
    public static void parseS3UriExample(S3Client s3Client, String s3ObjectUrl) {
        logger.info(s3ObjectUrl);
        // Console output:
        // 'https://s3.us-west-1.amazonaws.com/myBucket/resources/doc.txt?
versionId=abc123&partNumber=77&partNumber=88'.

        // Create an S3Utilities object using the configuration of the s3Client.
        S3Utilities s3Utilities = s3Client.utilities();

        // From a String URL create a URI object to pass to the parseUri() method.
        URI uri = URI.create(s3ObjectUrl);
        S3Uri s3Uri = s3Utilities.parseUri(uri);

        // If the URI contains no value for the Region, bucket or key, the SDK
returns
        // an empty Optional.
        // The SDK returns decoded URI values.

        Region region = s3Uri.region().orElse(null);
        log("region", region);
        // Console output: 'region: us-west-1'.

        String bucket = s3Uri.bucket().orElse(null);
        log("bucket", bucket);
        // Console output: 'bucket: myBucket'.

        String key = s3Uri.key().orElse(null);
        log("key", key);
        // Console output: 'key: resources/doc.txt'.

        Boolean isPathStyle = s3Uri.isPathStyle();
        log("isPathStyle", isPathStyle);
        // Console output: 'isPathStyle: true'.

        // If the URI contains no query parameters, the SDK returns an empty map.
        Map<String, List<String>> queryParams = s3Uri.rawQueryParameters();
        log("rawQueryParameters", queryParams);
        // Console output: 'rawQueryParameters: {versionId=[abc123], partNumber=[77,
// 88]}'

        // Retrieve the first or all values for a query parameter as shown in the
// following code.

```

```

    String versionId =
s3Uri.firstMatchingRawQueryParameter("versionId").orElse(null);
    log("firstMatchingRawQueryParameter-versionId", versionId);
    // Console output: 'firstMatchingRawQueryParameter-versionId: abc123'.

    String partNumber =
s3Uri.firstMatchingRawQueryParameter("partNumber").orElse(null);
    log("firstMatchingRawQueryParameter-partNumber", partNumber);
    // Console output: 'firstMatchingRawQueryParameter-partNumber: 77'.

    List<String> partNumbers =
s3Uri.firstMatchingRawQueryParameters("partNumber");
    log("firstMatchingRawQueryParameter", partNumbers);
    // Console output: 'firstMatchingRawQueryParameter: [77, 88]'.

    /*
    * Object keys and query parameters with reserved or unsafe characters, must
be
    * URL-encoded.
    * For example replace whitespace " " with "%20".
    * Valid:
    * "https://s3.us-west-1.amazonaws.com/myBucket/object%20key?query=
%5Bbrackets%5D"
    * Invalid:
    * "https://s3.us-west-1.amazonaws.com/myBucket/object key?query=[brackets]"
    *
    * Virtual-hosted-style URIs with bucket names that contain a dot, ".", the
dot
    * must not be URL-encoded.
    * Valid: "https://my.Bucket.s3.us-west-1.amazonaws.com/key"
    * Invalid: "https://my%2EBucket.s3.us-west-1.amazonaws.com/key"
    */
}

private static void log(String s3UriElement, Object element) {
    if (element == null) {
        logger.info("{}: {}", s3UriElement, "null");
    } else {
        logger.info("{}: {}", s3UriElement, element);
    }
}
}

```

## Esegui un caricamento in più parti

L'esempio di codice seguente mostra come eseguire un caricamento in più parti in un oggetto Amazon S3.

### SDK per Java 2.x

#### Note

C'è dell'altro GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Gli esempi di codice utilizzano le seguenti importazioni.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.exception.SdkException;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.services.s3.S3AsyncClient;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.CompletedMultipartUpload;
import software.amazon.awssdk.services.s3.model.CompletedPart;
import software.amazon.awssdk.services.s3.model.CreateMultipartUploadResponse;
import software.amazon.awssdk.services.s3.model.PutObjectResponse;
import software.amazon.awssdk.services.s3.model.UploadPartRequest;
import software.amazon.awssdk.services.s3.model.UploadPartResponse;
import software.amazon.awssdk.services.s3.waiters.S3Waiter;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.FileUpload;
import software.amazon.awssdk.transfer.s3.model.UploadFileRequest;

import java.io.IOException;
import java.io.RandomAccessFile;
import java.net.URISyntaxException;
import java.net.URL;
import java.nio.ByteBuffer;
import java.nio.file.Paths;
import java.util.ArrayList;
import java.util.List;
import java.util.Objects;
import java.util.UUID;
import java.util.concurrent.CompletableFuture;
```

Utilizza [S3 Transfer Manager](#) sul [client S3 basato su CRT AWS](#) per eseguire in modo trasparente un caricamento in più parti quando le dimensioni del contenuto superano una soglia. Le dimensioni soglia predefinite sono di 8 MB.

```
public void multipartUploadWithTransferManager(String filePath) {
    S3TransferManager transferManager = S3TransferManager.create();
    UploadFileRequest uploadFileRequest = UploadFileRequest.builder()
        .putObjectRequest(b -> b
            .bucket(bucketName)
            .key(key))
        .source(Paths.get(filePath))
        .build();
    FileUpload fileUpload = transferManager.uploadFile(uploadFileRequest);
    fileUpload.completionFuture().join();
    transferManager.close();
}
```

Usa l'[API S3Client](#) per eseguire un caricamento in più parti.

```
public void multipartUploadWithS3Client(String filePath) {

    // Initiate the multipart upload.
    CreateMultipartUploadResponse createMultipartUploadResponse =
s3Client.createMultipartUpload(b -> b
        .bucket(bucketName)
        .key(key));
    String uploadId = createMultipartUploadResponse.uploadId();

    // Upload the parts of the file.
    int partNumber = 1;
    List<CompletedPart> completedParts = new ArrayList<>();
    ByteBuffer bb = ByteBuffer.allocate(1024 * 1024 * 5); // 5 MB byte buffer

    try (RandomAccessFile file = new RandomAccessFile(filePath, "r")) {
        long fileSize = file.length();
        long position = 0;
        while (position < fileSize) {
            file.seek(position);
            long read = file.getChannel().read(bb);
```

```
        bb.flip(); // Swap position and limit before reading from the
buffer.

        UploadPartRequest uploadPartRequest = UploadPartRequest.builder()
            .bucket(bucketName)
            .key(key)
            .uploadId(uploadId)
            .partNumber(partNumber)
            .build();

        UploadPartResponse partResponse = s3Client.uploadPart(
            uploadPartRequest,
            RequestBody.fromByteBuffer(bb));

        CompletedPart part = CompletedPart.builder()
            .partNumber(partNumber)
            .eTag(partResponse.eTag())
            .build();
        completedParts.add(part);

        bb.clear();
        position += read;
        partNumber++;
    }
} catch (IOException e) {
    logger.error(e.getMessage());
}

// Complete the multipart upload.
s3Client.completeMultipartUpload(b -> b
    .bucket(bucketName)
    .key(key)
    .uploadId(uploadId)

.multipartUpload(CompletedMultipartUpload.builder().parts(completedParts).build()));
}
```

Utilizza l'[AsyncClient API S3](#) con il supporto multiparte abilitato per eseguire un caricamento in più parti.

```
public void multipartUploadWithS3AsyncClient(String filePath) {
```



```
// Enable multipart support.
S3AsyncClient s3AsyncClient = S3AsyncClient.builder()
    .multipartEnabled(true)
    .build();

CompletableFuture<PutObjectResponse> response = s3AsyncClient.putObject(b ->
b
    .bucket(bucketName)
    .key(key),
    Paths.get(filePath));

response.join();
logger.info("File uploaded in multiple 8 MiB parts using S3AsyncClient.");
}
```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella Documentazione di riferimento delle API AWS SDK for Java 2.x .
  - [CompleteMultipartUpload](#)
  - [CreateMultipartUpload](#)
  - [UploadPart](#)

Tieni traccia dei caricamenti e dei download

Il seguente esempio di codice mostra come tenere traccia del caricamento o del download di un oggetto Amazon S3.

SDK per Java 2.x

#### Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Tieni traccia dello stato di avanzamento del caricamento di un file.

```
public void trackUploadFile(S3TransferManager transferManager, String
bucketName,
String key, URI filePathURI) {
```

```

UploadFileRequest uploadFileRequest = UploadFileRequest.builder()
    .putObjectRequest(b -> b.bucket(bucketName).key(key))
    .addTransferListener(LoggingTransferListener.create()) // Add
listener.
    .source(Paths.get(filePathURI))
    .build();

```

```
FileUpload fileUpload = transferManager.uploadFile(uploadFileRequest);
```

```
fileUpload.completionFuture().join();
```

```
/*
```

The SDK provides a `LoggingTransferListener` implementation of the `TransferListener` interface.

You can also implement the interface to provide your own logic.

Configure `log4j2` with settings such as the following.

```

<Configuration status="WARN">
  <Appenders>
    <Console name="AlignedConsoleAppender" target="SYSTEM_OUT">
      <PatternLayout pattern="%m%n"/>
    </Console>
  </Appenders>

  <Loggers>
    <logger
name="software.amazon.awssdk.transfer.s3.progress.LoggingTransferListener"
level="INFO" additivity="false">
      <AppenderRef ref="AlignedConsoleAppender"/>
    </logger>
  </Loggers>
</Configuration>

```

`Log4j2` logs the progress. The following is example output for a 21.3 MB file upload.

```

Transfer initiated...
|                               | 0.0%
|====                           | 21.1%
|=====                         | 60.5%
|=====                         | 100.0%
Transfer complete!

```

```
*/
```

```
}
```

## Tieni traccia dello stato di avanzamento del download di un file.

```
public void trackDownloadFile(S3TransferManager transferManager, String
bucketName,
                            String key, String downloadedFilePath) {
    DownloadFileRequest downloadFileRequest = DownloadFileRequest.builder()
        .getObjectRequest(b -> b.bucket(bucketName).key(key))
        .addTransferListener(LoggingTransferListener.create()) // Add
listener.
        .destination(Paths.get(downloadedFilePath))
        .build();
```

```
    FileDownload downloadFile =
transferManager.downloadFile(downloadFileRequest);
```

```
    CompletedFileDownload downloadResult =
downloadFile.completionFuture().join();
```

```
    /*
```

The SDK provides a `LoggingTransferListener` implementation of the `TransferListener` interface.

You can also implement the interface to provide your own logic.

Configure log4j2 with settings such as the following.

```
<Configuration status="WARN">
  <Appenders>
    <Console name="AlignedConsoleAppender" target="SYSTEM_OUT">
      <PatternLayout pattern="%m%n"/>
    </Console>
  </Appenders>

  <Loggers>
    <logger
name="software.amazon.awssdk.transfer.s3.progress.LoggingTransferListener"
level="INFO" additivity="false">
      <AppenderRef ref="AlignedConsoleAppender"/>
    </logger>
  </Loggers>
</Configuration>
```

Log4j2 logs the progress. The following is example output for a 21.3 MB file download.

```
Transfer initiated...
|=====          | 39.4%
|=====          | 78.8%
```

```

                |=====| 100.0%
                Transfer complete!
            */
        }

```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella Documentazione di riferimento delle API AWS SDK for Java 2.x .
  - [GetObject](#)
  - [PutObject](#)

### Caricare una directory in un bucket

L'esempio di codice seguente mostra come caricare una directory locale in modo ricorsivo in un bucket Amazon Simple Storage Service (Amazon S3).

### SDK per Java 2.x

#### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Usa un [S3 TransferManager](#) per [caricare una directory locale](#). Visualizza il [file completo](#) ed esegui il [test](#).

```

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.s3.model.ObjectIdentifier;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.CompletedDirectoryUpload;
import software.amazon.awssdk.transfer.s3.model.DirectoryUpload;
import software.amazon.awssdk.transfer.s3.model.UploadDirectoryRequest;

import java.net.URI;
import java.net.URISyntaxException;
import java.net.URL;
import java.nio.file.Paths;
import java.util.UUID;

```

```
public Integer uploadDirectory(S3TransferManager transferManager,
    URI sourceDirectory, String bucketName) {
    DirectoryUpload directoryUpload =
transferManager.uploadDirectory(UploadDirectoryRequest.builder()
    .source(Paths.get(sourceDirectory))
    .bucket(bucketName)
    .build());

    CompletedDirectoryUpload completedDirectoryUpload =
directoryUpload.completionFuture().join();
    completedDirectoryUpload.failedTransfers()
        .forEach(fail -> logger.warn("Object [{}] failed to transfer",
fail.toString()));
    return completedDirectoryUpload.failedTransfers().size();
}
```

- Per i dettagli sull'API, consulta la sezione [UploadDirectory AWS SDK for Java 2.xAPI Reference](#).

## Caricamento o download di file di grandi dimensioni

Il seguente esempio di codice mostra come caricare o scaricare file di grandi dimensioni da e verso Amazon S3.

Per ulteriori informazioni, consulta [Caricamento di un oggetto utilizzando il caricamento in più parti](#).

## SDK per Java 2.x

### Note

C'è altro su GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Richiama le funzioni che trasferiscono file da e verso un bucket S3 utilizzando S3.  
TransferManager

```
public Integer downloadObjectsToDirectory(S3TransferManager transferManager,
    URI destinationPathURI, String bucketName) {
```

```

        DirectoryDownload directoryDownload =
transferManager.downloadDirectory(DownloadDirectoryRequest.builder()
        .destination(Paths.get(destinationPathURI))
        .bucket(bucketName)
        .build());
        CompletedDirectoryDownload completedDirectoryDownload =
directoryDownload.completionFuture().join();

        completedDirectoryDownload.failedTransfers()
        .forEach(fail -> logger.warn("Object [{}] failed to transfer",
fail.toString()));
        return completedDirectoryDownload.failedTransfers().size();
    }

```

### Caricamento di un'intera directory locale.

```

public Integer uploadDirectory(S3TransferManager transferManager,
        URI sourceDirectory, String bucketName) {
    DirectoryUpload directoryUpload =
transferManager.uploadDirectory(UploadDirectoryRequest.builder()
        .source(Paths.get(sourceDirectory))
        .bucket(bucketName)
        .build());

    CompletedDirectoryUpload completedDirectoryUpload =
directoryUpload.completionFuture().join();
    completedDirectoryUpload.failedTransfers()
        .forEach(fail -> logger.warn("Object [{}] failed to transfer",
fail.toString()));
    return completedDirectoryUpload.failedTransfers().size();
}

```

### Caricamento di un singolo file.

```

public String uploadFile(S3TransferManager transferManager, String bucketName,
        String key, URI filePathURI) {
    UploadFileRequest uploadFileRequest = UploadFileRequest.builder()
        .putObjectRequest(b -> b.bucket(bucketName).key(key))
        .source(Paths.get(filePathURI))
        .build();
}

```

```
FileUpload fileUpload = transferManager.uploadFile(uploadFileRequest);

CompletedFileUpload uploadResult = fileUpload.completionFuture().join();
return uploadResult.response().eTag();
}
```

## Caricamento di un flusso di dimensioni sconosciute

Gli esempi di codice seguenti mostrano come caricare un flusso di dimensioni sconosciute in un oggetto Amazon S3.

### SDK per Java 2.x

#### Note

C'è altro da fare. [GitHub](#) Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Usa il [Client S3 basato su CRT AWS](#).

```
import com.example.s3.util.AsyncExampleUtils;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.async.AsyncRequestBody;
import software.amazon.awssdk.core.async.BlockingInputStreamAsyncRequestBody;
import software.amazon.awssdk.core.exception.SdkException;
import software.amazon.awssdk.services.s3.S3AsyncClient;
import software.amazon.awssdk.services.s3.model.PutObjectResponse;

import java.io.ByteArrayInputStream;
import java.util.UUID;
import java.util.concurrent.CompletableFuture;

/**
 * @param s3CrtAsyncClient - To upload content from a stream of unknown size,
 * use the AWS CRT-based S3 client. For more information, see
 * https://docs.aws.amazon.com/sdk-for-java/latest/
 * developer-guide/crt-based-s3-client.html.
 * @param bucketName - The name of the bucket.
```

```
* @param key - The name of the object.
* @return software.amazon.awssdk.services.s3.model.PutObjectResponse - Returns
metadata pertaining to the put object operation.
*/
public PutObjectResponse putObjectFromStream(S3AsyncClient s3CrtAsyncClient,
String bucketName, String key) {

    BlockingInputStreamAsyncRequestBody body =
        AsyncRequestBody.forBlockingInputStream(null); // 'null' indicates a
stream will be provided later.

    CompletableFuture<PutObjectResponse> responseFuture =
        s3CrtAsyncClient.putObject(r -> r.bucket(bucketName).key(key),
body);

    // AsyncExampleUtils.randomString() returns a random string up to 100
characters.
    String randomString = AsyncExampleUtils.randomString();
    logger.info("random string to upload: {}: length={}", randomString,
randomString.length());

    // Provide the stream of data to be uploaded.
    body.writeInputStream(new ByteArrayInputStream(randomString.getBytes()));

    PutObjectResponse response = responseFuture.join(); // Wait for the
response.
    logger.info("Object {} uploaded to bucket {}.", key, bucketName);
    return response;
}
}
```

## Usa [Amazon S3 Transfer Manager](#).

```
import com.example.s3.util.AsyncExampleUtils;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.async.AsyncRequestBody;
import software.amazon.awssdk.core.async.BlockingInputStreamAsyncRequestBody;
import software.amazon.awssdk.core.exception.SdkException;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.CompletedUpload;
import software.amazon.awssdk.transfer.s3.model.Upload;
```



```
import java.io.ByteArrayInputStream;
import java.util.UUID;

/**
 * @param transferManager - To upload content from a stream of unknown size, use
 the S3TransferManager based on the AWS CRT-based S3 client.
 *
 * For more information, see https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/transfer-manager.html.
 * @param bucketName - The name of the bucket.
 * @param key - The name of the object.
 * @return - software.amazon.awssdk.transfer.s3.model.CompletedUpload - The
 result of the completed upload.
 */
public CompletedUpload uploadStream(S3TransferManager transferManager, String
bucketName, String key) {

    BlockingInputStreamAsyncRequestBody body =
        AsyncRequestBody.forBlockingInputStream(null); // 'null' indicates a
stream will be provided later.

    Upload upload = transferManager.upload(builder -> builder
        .requestBody(body)
        .putObjectRequest(req -> req.bucket(bucketName).key(key))
        .build());

    // AsyncExampleUtils.randomString() returns a random string up to 100
characters.
    String randomString = AsyncExampleUtils.randomString();
    logger.info("random string to upload: {}: length={}", randomString,
randomString.length());

    // Provide the stream of data to be uploaded.
    body.writeInputStream(new ByteArrayInputStream(randomString.getBytes()));

    return upload.completionFuture().join();
}
}
```

## Utilizzo dei checksum

L'esempio di codice seguente mostra come utilizzare i checksum per lavorare con un oggetto Amazon S3.

### SDK per Java 2.x

#### Note

C'è dell'altro GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Gli esempi di codice utilizzano un sottoinsieme delle seguenti importazioni.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.exception.SdkException;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.ChecksumAlgorithm;
import software.amazon.awssdk.services.s3.model.ChecksumMode;
import software.amazon.awssdk.services.s3.model.CompletedMultipartUpload;
import software.amazon.awssdk.services.s3.model.CompletedPart;
import software.amazon.awssdk.services.s3.model.CreateMultipartUploadResponse;
import software.amazon.awssdk.services.s3.model.GetObjectResponse;
import software.amazon.awssdk.services.s3.model.UploadPartRequest;
import software.amazon.awssdk.services.s3.model.UploadPartResponse;
import software.amazon.awssdk.services.s3.waiters.S3Waiter;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.FileUpload;
import software.amazon.awssdk.transfer.s3.model.UploadFileRequest;

import java.io.FileInputStream;
import java.io.IOException;
import java.io.RandomAccessFile;
import java.net.URISyntaxException;
import java.net.URL;
import java.nio.ByteBuffer;
import java.nio.file.Paths;
import java.security.DigestInputStream;
import java.security.MessageDigest;
```

```
import java.security.NoSuchAlgorithmException;
import java.util.ArrayList;
import java.util.Base64;
import java.util.List;
import java.util.Objects;
import java.util.UUID;
```

Specifica un algoritmo di checksum per il metodo `putObject` quando [crei il `PutObjectRequest`](#).

```
public void putObjectWithChecksum() {
    s3Client.putObject(b -> b
        .bucket(bucketName)
        .key(key)
        .checksumAlgorithm(ChecksumAlgorithm.CRC32),
        RequestBody.fromString("This is a test"));
}
```

Verifica il checksum per il `getObject` metodo quando [crei il `GetObjectRequest`](#).

```
public GetObjectResponse getObjectWithChecksum() {
    return s3Client.getObject(b -> b
        .bucket(bucketName)
        .key(key)
        .checksumMode(ChecksumMode.ENABLED))
        .response();
}
```

Precalcola un checksum per il metodo `putObject` quando [crei il `PutObjectRequest`](#).

```
public void putObjectWithPrecalculatedChecksum(String filePath) {
    String checksum = calculateChecksum(filePath, "SHA-256");

    s3Client.putObject((b -> b
        .bucket(bucketName)
        .key(key)
        .checksumSHA256(checksum)),
        RequestBody.fromFile(Paths.get(filePath)));
}
```

Utilizza [S3 Transfer Manager](#) sul [client S3 basato su CRT AWS](#) per eseguire in modo trasparente un caricamento in più parti quando le dimensioni del contenuto superano una soglia. Le dimensioni soglia predefinite sono di 8 MB.

Puoi specificare un algoritmo di checksum da utilizzare nell'SDK. Per impostazione predefinita, l'SDK utilizza l'algoritmo CRC32.

```
public void multipartUploadWithChecksumTm(String filePath) {
    S3TransferManager transferManager = S3TransferManager.create();
    UploadFileRequest uploadFileRequest = UploadFileRequest.builder()
        .putObjectRequest(b -> b
            .bucket(bucketName)
            .key(key)
            .checksumAlgorithm(ChecksumAlgorithm.SHA1))
        .source(Paths.get(filePath))
        .build();
    FileUpload fileUpload = transferManager.uploadFile(uploadFileRequest);
    fileUpload.completionFuture().join();
    transferManager.close();
}
```

Utilizza l'API [S3Client o \(API S3\)](#) per eseguire un AsyncClient caricamento in più parti. Se specifichi un checksum aggiuntivo, devi specificare l'algoritmo da utilizzare all'avvio del caricamento. Inoltre, devi specificare l'algoritmo per ogni richiesta parte e fornire il checksum calcolato per ciascuna parte dopo che è stata caricata.

```
public void multipartUploadWithChecksumS3Client(String filePath) {
    ChecksumAlgorithm algorithm = ChecksumAlgorithm.CRC32;

    // Initiate the multipart upload.
    CreateMultipartUploadResponse createMultipartUploadResponse =
s3Client.createMultipartUpload(b -> b
        .bucket(bucketName)
        .key(key)
        .checksumAlgorithm(algorithm)); // Checksum specified on initiation.
    String uploadId = createMultipartUploadResponse.uploadId();

    // Upload the parts of the file.
    int partNumber = 1;
```

```

List<CompletedPart> completedParts = new ArrayList<>();
ByteBuffer bb = ByteBuffer.allocate(1024 * 1024 * 5); // 5 MB byte buffer

try (RandomAccessFile file = new RandomAccessFile(filePath, "r")) {
    long fileSize = file.length();
    long position = 0;
    while (position < fileSize) {
        file.seek(position);
        long read = file.getChannel().read(bb);

        bb.flip(); // Swap position and limit before reading from the
buffer.

        UploadPartRequest uploadPartRequest = UploadPartRequest.builder()
            .bucket(bucketName)
            .key(key)
            .uploadId(uploadId)
            .checksumAlgorithm(algorithm) // Checksum specified on each
part.

            .partNumber(partNumber)
            .build();

        UploadPartResponse partResponse = s3Client.uploadPart(
            uploadPartRequest,
            RequestBody.fromByteBuffer(bb));

        CompletedPart part = CompletedPart.builder()
            .partNumber(partNumber)
            .checksumCRC32(partResponse.checksumCRC32()) // Provide the
calculated checksum.

            .eTag(partResponse.eTag())
            .build();
        completedParts.add(part);

        bb.clear();
        position += read;
        partNumber++;
    }
} catch (IOException e) {
    System.err.println(e.getMessage());
}

// Complete the multipart upload.
s3Client.completeMultipartUpload(b -> b
    .bucket(bucketName)

```

```
        .key(key)
        .uploadId(uploadId)

    .multipartUpload(CompletedMultipartUpload.builder().parts(completedParts).build()));
    }
```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella Documentazione di riferimento delle API AWS SDK for Java 2.x .
  - [CompleteMultipartUpload](#)
  - [CreateMultipartUpload](#)
  - [UploadPart](#)

## Esempi serverless

Richiamo di una funzione Lambda da un trigger Amazon S3

Il seguente esempio di codice mostra come implementare una funzione Lambda che riceve un evento attivato dal caricamento di un oggetto in un bucket S3. La funzione recupera il nome del bucket S3 e la chiave dell'oggetto dal parametro evento e chiama l'API Amazon S3 per recuperare e registrare il tipo di contenuto dell'oggetto.

SDK per Java 2.x

### Note

C'è di più su. [GitHub Trova l'esempio completo](#) e scopri come eseguire la configurazione e l'esecuzione nel repository di [Esempi serverless](#).

Utilizzo di un evento S3 con Lambda tramite Java.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package example;

import software.amazon.awssdk.services.s3.model.HeadObjectRequest;
import software.amazon.awssdk.services.s3.model.HeadObjectResponse;
import software.amazon.awssdk.services.s3.S3Client;
```

```
import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.S3Event;
import
    com.amazonaws.services.lambda.runtime.events.models.s3.S3EventNotification.S3EventNotificat

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

public class Handler implements RequestHandler<S3Event, String> {
    private static final Logger logger = LoggerFactory.getLogger(Handler.class);
    @Override
    public String handleRequest(S3Event s3event, Context context) {
        try {
            S3EventNotificationRecord record = s3event.getRecords().get(0);
            String srcBucket = record.getS3().getBucket().getName();
            String srcKey = record.getS3().getObject().getUrlDecodedKey();

            S3Client s3Client = S3Client.builder().build();
            HeadObjectResponse headObject = getHeadObject(s3Client, srcBucket,
srcKey);

            logger.info("Successfully retrieved " + srcBucket + "/" + srcKey + " of
type " + headObject.contentType());

            return "Ok";
        } catch (Exception e) {
            throw new RuntimeException(e);
        }
    }

    private HeadObjectResponse getHeadObject(S3Client s3Client, String bucket,
String key) {
        HeadObjectRequest headObjectRequest = HeadObjectRequest.builder()
            .bucket(bucket)
            .key(key)
            .build();
        return s3Client.headObject(headObjectRequest);
    }
}
```

## Esempi di S3 Glacier con SDK for Java 2.x

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando S3 Glacier AWS SDK for Java 2.x .

Le operazioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le operazioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati e negli esempi tra servizi.

Scenari: esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio.

Ogni esempio include un collegamento a GitHub, dove è possibile trovare istruzioni su come configurare ed eseguire il codice nel contesto.

### Argomenti

- [Azioni](#)

### Azioni

#### CreateVault

Il seguente esempio di codice mostra come utilizzare `CreateVault`.

#### SDK per Java 2.x

#### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.glacier.GlacierClient;
import software.amazon.awssdk.services.glacier.model.CreateVaultRequest;
import software.amazon.awssdk.services.glacier.model.CreateVaultResponse;
import software.amazon.awssdk.services.glacier.model.GlacierException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
```



```
*
* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class CreateVault {
    public static void main(String[] args) {
        final String usage = ""

            Usage:    <vaultName>

            Where:
                vaultName - The name of the vault to create.

            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String vaultName = args[0];
        GlacierClient glacier = GlacierClient.builder()
            .region(Region.US_EAST_1)
            .build();

        createGlacierVault(glacier, vaultName);
        glacier.close();
    }

    public static void createGlacierVault(GlacierClient glacier, String vaultName) {
        try {
            CreateVaultRequest vaultRequest = CreateVaultRequest.builder()
                .vaultName(vaultName)
                .build();

            CreateVaultResponse createVaultResult =
glacier.createVault(vaultRequest);
            System.out.println("The URI of the new vault is " +
createVaultResult.location());

        } catch (GlacierException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

```

    }
  }
}

```

- Per i dettagli sull'API, consulta la [CreateVault](#) sezione AWS SDK for Java 2.x API Reference.

## DeleteArchive

Il seguente esempio di codice mostra come utilizzare `DeleteArchive`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.glacier.GlacierClient;
import software.amazon.awssdk.services.glacier.model.DeleteArchiveRequest;
import software.amazon.awssdk.services.glacier.model.GlacierException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteArchive {
    public static void main(String[] args) {
        final String usage = ""

                Usage:    <vaultName> <accountId> <archiveId>

                Where:
                    vaultName - The name of the vault that contains the archive to
delete.

```

```
        accountId - The account ID value.
        archiveId - The archive ID value.
        """;

    if (args.length != 3) {
        System.out.println(usage);
        System.exit(1);
    }

    String vaultName = args[0];
    String accountId = args[1];
    String archiveId = args[2];
    GlacierClient glacier = GlacierClient.builder()
        .region(Region.US_EAST_1)
        .build();

    deleteGlacierArchive(glacier, vaultName, accountId, archiveId);
    glacier.close();
}

public static void deleteGlacierArchive(GlacierClient glacier, String vaultName,
String accountId,
    String archiveId) {
    try {
        DeleteArchiveRequest delArcRequest = DeleteArchiveRequest.builder()
            .vaultName(vaultName)
            .accountId(accountId)
            .archiveId(archiveId)
            .build();

        glacier.deleteArchive(delArcRequest);
        System.out.println("The archive was deleted.");

    } catch (GlacierException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Per i dettagli sull'API, consulta la [DeleteArchive](#) sezione AWS SDK for Java 2.x API Reference.

## DeleteVault

Il seguente esempio di codice mostra come utilizzare DeleteVault.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.glacier.GlacierClient;
import software.amazon.awssdk.services.glacier.model.DeleteVaultRequest;
import software.amazon.awssdk.services.glacier.model.GlacierException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteVault {
    public static void main(String[] args) {

        final String usage = ""

            Usage:    <vaultName>

            Where:
                vaultName - The name of the vault to delete.\s
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String vaultName = args[0];
        GlacierClient glacier = GlacierClient.builder()
```

```
        .region(Region.US_EAST_1)
        .build();

    deleteGlacierVault(glacier, vaultName);
    glacier.close();
}

public static void deleteGlacierVault(GlacierClient glacier, String vaultName) {
    try {
        DeleteVaultRequest delVaultRequest = DeleteVaultRequest.builder()
            .vaultName(vaultName)
            .build();

        glacier.deleteVault(delVaultRequest);
        System.out.println("The vault was deleted!");

    } catch (GlacierException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Per i dettagli sull'API, consulta la [DeleteVault](#) sezione AWS SDK for Java 2.x API Reference.

## InitiateJob

Il seguente esempio di codice mostra come utilizzare `InitiateJob`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Recupera un inventario del caveau.

```
import software.amazon.awssdk.core.ResponseBytes;
import software.amazon.awssdk.regions.Region;
```

```
import software.amazon.awssdk.services.glacier.GlacierClient;
import software.amazon.awssdk.services.glacier.model.JobParameters;
import software.amazon.awssdk.services.glacier.model.InitiateJobResponse;
import software.amazon.awssdk.services.glacier.model.GlacierException;
import software.amazon.awssdk.services.glacier.model.InitiateJobRequest;
import software.amazon.awssdk.services.glacier.model.DescribeJobRequest;
import software.amazon.awssdk.services.glacier.model.DescribeJobResponse;
import software.amazon.awssdk.services.glacier.model.GetJobOutputRequest;
import software.amazon.awssdk.services.glacier.model.GetJobOutputResponse;
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.OutputStream;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ArchiveDownload {
    public static void main(String[] args) {

        final String usage = ""

            Usage:    <vaultName> <accountId> <path>

            Where:
                vaultName - The name of the vault.
                accountId - The account ID value.
                path - The path where the file is written to.
            """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String vaultName = args[0];
        String accountId = args[1];
        String path = args[2];
        GlacierClient glacier = GlacierClient.builder()
```

```
        .region(Region.US_EAST_1)
        .build();

    String jobNum = createJob(glacier, vaultName, accountId);
    checkJob(glacier, jobNum, vaultName, accountId, path);
    glacier.close();
}

public static String createJob(GlacierClient glacier, String vaultName, String
accountId) {
    try {
        JobParameters job = JobParameters.builder()
            .type("inventory-retrieval")
            .build();

        InitiateJobRequest initJob = InitiateJobRequest.builder()
            .jobParameters(job)
            .accountId(accountId)
            .vaultName(vaultName)
            .build();

        InitiateJobResponse response = glacier.initiateJob(initJob);
        System.out.println("The job ID is: " + response.jobId());
        System.out.println("The relative URI path of the job is: " +
response.location());
        return response.jobId();

    } catch (GlacierException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }

    return "";
}

// Poll S3 Glacier = Polling a Job may take 4-6 hours according to the
// Documentation.
public static void checkJob(GlacierClient glacier, String jobId, String name,
String account, String path) {
    try {
        boolean finished = false;
        String jobStatus;
        int yy = 0;
    }
}
```

```
while (!finished) {
    DescribeJobRequest jobRequest = DescribeJobRequest.builder()
        .jobId(jobId)
        .accountId(account)
        .vaultName(name)
        .build();

    DescribeJobResponse response = glacier.describeJob(jobRequest);
    jobStatus = response.statusCodeAsString();

    if (jobStatus.compareTo("Succeeded") == 0)
        finished = true;
    else {
        System.out.println(yy + " status is: " + jobStatus);
        Thread.sleep(1000);
    }
    yy++;
}

System.out.println("Job has Succeeded");
GetJobOutputRequest jobOutputRequest = GetJobOutputRequest.builder()
    .jobId(jobId)
    .vaultName(name)
    .accountId(account)
    .build();

ResponseBytes<GetJobOutputResponse> objectBytes =
glacier.getJobOutputAsBytes(jobOutputRequest);
// Write the data to a local file.
byte[] data = objectBytes.asByteArray();
File myFile = new File(path);
OutputStream os = new FileOutputStream(myFile);
os.write(data);
System.out.println("Successfully obtained bytes from a Glacier vault");
os.close();

} catch (GlacierException | InterruptedException | IOException e) {
    System.out.println(e.getMessage());
    System.exit(1);
}
}
```



- Per i dettagli sull'API, consulta la sezione AWS SDK for Java 2.x API [InitiateJobReference](#).

## ListVaults

Il seguente esempio di codice mostra come utilizzare `ListVaults`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.glacier.model.ListVaultsRequest;
import software.amazon.awssdk.services.glacier.model.ListVaultsResponse;
import software.amazon.awssdk.services.glacier.GlacierClient;
import software.amazon.awssdk.services.glacier.model.DescribeVaultOutput;
import software.amazon.awssdk.services.glacier.model.GlacierException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListVaults {
    public static void main(String[] args) {
        GlacierClient glacier = GlacierClient.builder()
            .region(Region.US_EAST_1)
            .build();

        listAllVault(glacier);
        glacier.close();
    }
}
```

```
public static void listAllVault(GlacierClient glacier) {
    boolean listComplete = false;
    String newMarker = null;
    int totalVaults = 0;
    System.out.println("Your Amazon Glacier vaults:");
    try {
        while (!listComplete) {
            ListVaultsResponse response = null;
            if (newMarker != null) {
                ListVaultsRequest request = ListVaultsRequest.builder()
                    .marker(newMarker)
                    .build();

                response = glacier.listVaults(request);
            } else {
                ListVaultsRequest request = ListVaultsRequest.builder()
                    .build();
                response = glacier.listVaults(request);
            }

            List<DescribeVaultOutput> vaultList = response.vaultList();
            for (DescribeVaultOutput v : vaultList) {
                totalVaults += 1;
                System.out.println("* " + v.vaultName());
            }

            // Check for further results.
            newMarker = response.marker();
            if (newMarker == null) {
                listComplete = true;
            }
        }

        if (totalVaults == 0) {
            System.out.println("No vaults found.");
        }

    } catch (GlacierException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Per i dettagli sull'API, consulta la [ListVaults](#) sezione AWS SDK for Java 2.x API Reference.

## UploadArchive

Il seguente esempio di codice mostra come utilizzare `UploadArchive`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.glacier.GlacierClient;
import software.amazon.awssdk.services.glacier.model.UploadArchiveRequest;
import software.amazon.awssdk.services.glacier.model.UploadArchiveResponse;
import software.amazon.awssdk.services.glacier.model.GlacierException;
import java.io.File;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.io.FileInputStream;
import java.io.IOException;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class UploadArchive {

    static final int ONE_MB = 1024 * 1024;

    public static void main(String[] args) {
```

```
final String usage = ""

Usage:  <strPath> <vaultName>\s

Where:
    strPath - The path to the archive to upload (for example, C:\\AWS
\\test.pdf).
    vaultName - The name of the vault.
"";

if (args.length != 2) {
    System.out.println(usage);
    System.exit(1);
}

String strPath = args[0];
String vaultName = args[1];
File myFile = new File(strPath);
Path path = Paths.get(strPath);
GlacierClient glacier = GlacierClient.builder()
    .region(Region.US_EAST_1)
    .build();

String archiveId = uploadContent(glacier, path, vaultName, myFile);
System.out.println("The ID of the archived item is " + archiveId);
glacier.close();
}

public static String uploadContent(GlacierClient glacier, Path path, String
vaultName, File myFile) {
    // Get an SHA-256 tree hash value.
    String checkVal = computeSHA256(myFile);
    try {
        UploadArchiveRequest uploadRequest = UploadArchiveRequest.builder()
            .vaultName(vaultName)
            .checksum(checkVal)
            .build();

        UploadArchiveResponse res = glacier.uploadArchive(uploadRequest, path);
        return res.archiveId();
    } catch (GlacierException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

```
    }
    return "";
}

private static String computeSHA256(File inputFile) {
    try {
        byte[] treeHash = computeSHA256TreeHash(inputFile);
        System.out.printf("SHA-256 tree hash = %s\n", toHex(treeHash));
        return toHex(treeHash);

    } catch (IOException ioe) {
        System.err.format("Exception when reading from file %s: %s", inputFile,
ioe.getMessage());
        System.exit(-1);

    } catch (NoSuchAlgorithmException nsae) {
        System.err.format("Cannot locate MessageDigest algorithm for SHA-256:
%s", nsae.getMessage());
        System.exit(-1);
    }
    return "";
}

public static byte[] computeSHA256TreeHash(File inputFile) throws IOException,
    NoSuchAlgorithmException {

    byte[][] chunkSHA256Hashes = getChunkSHA256Hashes(inputFile);
    return computeSHA256TreeHash(chunkSHA256Hashes);
}

/**
 * Computes an SHA256 checksum for each 1 MB chunk of the input file. This
 * includes the checksum for the last chunk, even if it's smaller than 1 MB.
 */
public static byte[][] getChunkSHA256Hashes(File file) throws IOException,
    NoSuchAlgorithmException {

    MessageDigest md = MessageDigest.getInstance("SHA-256");
    long numChunks = file.length() / ONE_MB;
    if (file.length() % ONE_MB > 0) {
        numChunks++;
    }

    if (numChunks == 0) {
```

```

        return new byte[][] { md.digest() };
    }

    byte[][] chunkSHA256Hashes = new byte[(int) numChunks][];
    FileInputStream fileStream = null;

    try {
        fileStream = new FileInputStream(file);
        byte[] buff = new byte[ONE_MB];

        int bytesRead;
        int idx = 0;

        while ((bytesRead = fileStream.read(buff, 0, ONE_MB)) > 0) {
            md.reset();
            md.update(buff, 0, bytesRead);
            chunkSHA256Hashes[idx++] = md.digest();
        }

        return chunkSHA256Hashes;

    } finally {
        if (fileStream != null) {
            try {
                fileStream.close();
            } catch (IOException ioe) {
                System.err.printf("Exception while closing %s.\n %s",
file.getName(),
                                ioe.getMessage());
            }
        }
    }
}

/**
 * Computes the SHA-256 tree hash for the passed array of 1 MB chunk
 * checksums.
 */
public static byte[] computeSHA256TreeHash(byte[][] chunkSHA256Hashes)
    throws NoSuchAlgorithmException {

    MessageDigest md = MessageDigest.getInstance("SHA-256");
    byte[][] prevLvlHashes = chunkSHA256Hashes;
    while (prevLvlHashes.length > 1) {

```

```
int len = prevLvlHashes.length / 2;
if (prevLvlHashes.length % 2 != 0) {
    len++;
}

byte[][] currLvlHashes = new byte[len][];
int j = 0;
for (int i = 0; i < prevLvlHashes.length; i = i + 2, j++) {

    // If there are at least two elements remaining.
    if (prevLvlHashes.length - i > 1) {

        // Calculate a digest of the concatenated nodes.
        md.reset();
        md.update(prevLvlHashes[i]);
        md.update(prevLvlHashes[i + 1]);
        currLvlHashes[j] = md.digest();

    } else { // Take care of the remaining odd chunk
        currLvlHashes[j] = prevLvlHashes[i];
    }
}

prevLvlHashes = currLvlHashes;
}

return prevLvlHashes[0];
}

/**
 * Returns the hexadecimal representation of the input byte array
 */
public static String toHex(byte[] data) {
    StringBuilder sb = new StringBuilder(data.length * 2);
    for (byte datum : data) {
        String hex = Integer.toHexString(datum & 0xFF);

        if (hex.length() == 1) {
            // Append leading zero.
            sb.append("0");
        }
        sb.append(hex);
    }
    return sb.toString().toLowerCase();
}
```

```
}  
}
```

- Per i dettagli sull'API, consulta la [UploadArchive](#) sezione AWS SDK for Java 2.x API Reference.

## SageMaker esempi che utilizzano SDK for Java 2.x

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando AWS SDK for Java 2.x with SageMaker.

Le operazioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le operazioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati e negli esempi tra servizi.

Scenari: esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio.

Ogni esempio include un collegamento a GitHub, dove è possibile trovare istruzioni su come configurare ed eseguire il codice nel contesto.

Nozioni di base

Salve SageMaker

I seguenti esempi di codice mostrano come iniziare a utilizzare SageMaker.

SDK per Java 2.x

### Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
/**  
 * Before running this Java V2 code example, set up your development  
 * environment, including your credentials.  
 */
```



```
* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class HelloSageMaker {
    public static void main(String[] args) {
        Region region = Region.US_WEST_2;
        SageMakerClient sageMakerClient = SageMakerClient.builder()
            .region(region)
            .build();

        listBooks(sageMakerClient);
        sageMakerClient.close();
    }

    public static void listBooks(SageMakerClient sageMakerClient) {
        try {
            ListNotebookInstancesResponse notebookInstancesResponse =
sageMakerClient.listNotebookInstances();
            List<NotebookInstanceSummary> items =
notebookInstancesResponse.notebookInstances();
            for (NotebookInstanceSummary item : items) {
                System.out.println("The notebook name is: " +
item.notebookInstanceName());
            }

        } catch (SageMakerException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Per i dettagli sull'API, consulta la [ListNotebookInstances](#) sezione AWS SDK for Java 2.x API Reference.

## Argomenti

- [Azioni](#)
- [Scenari](#)

## Azioni

### CreatePipeline

Il seguente esempio di codice mostra come utilizzare `CreatePipeline`.

SDK per Java 2.x

#### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
// Create a pipeline from the example pipeline JSON.
public static void setupPipeline(SageMakerClient sageMakerClient, String
filePath, String roleArn,
    String functionArn, String pipelineName) {
    System.out.println("Setting up the pipeline.");
    JSONParser parser = new JSONParser();

    // Read JSON and get pipeline definition.
    try (FileReader reader = new FileReader(filePath)) {
        Object obj = parser.parse(reader);
        JSONObject jsonObject = (JSONObject) obj;
        JSONArray stepsArray = (JSONArray) jsonObject.get("Steps");
        for (Object stepObj : stepsArray) {
            JSONObject step = (JSONObject) stepObj;
            if (step.containsKey("FunctionArn")) {
                step.put("FunctionArn", functionArn);
            }
        }
        System.out.println(jsonObject);

        // Create the pipeline.
        CreatePipelineRequest pipelineRequest = CreatePipelineRequest.builder()
            .pipelineDescription("Java SDK example pipeline")
            .roleArn(roleArn)
            .pipelineName(pipelineName)
            .pipelineDefinition(jsonObject.toString())
            .build();

        sageMakerClient.createPipeline(pipelineRequest);
    }
}
```

```
    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    } catch (IOException | ParseException e) {
        throw new RuntimeException(e);
    }
}
```

- Per i dettagli sull'API, consulta la [CreatePipeline](#) sezione AWS SDK for Java 2.x API Reference.

## DeletePipeline

Il seguente esempio di codice mostra come utilizzare `DeletePipeline`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
// Delete a SageMaker pipeline by name.
public static void deletePipeline(SageMakerClient sageMakerClient, String
pipelineName) {
    DeletePipelineRequest pipelineRequest = DeletePipelineRequest.builder()
        .pipelineName(pipelineName)
        .build();

    sageMakerClient.deletePipeline(pipelineRequest);
    System.out.println("*** Successfully deleted " + pipelineName);
}
```

- Per i dettagli sull'API, consulta la [DeletePipeline](#) sezione AWS SDK for Java 2.x API Reference.

## DescribePipelineExecution

Il seguente esempio di codice mostra come utilizzare `DescribePipelineExecution`.

## SDK per Java 2.x

 Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
// Check the status of a pipeline execution.
public static void waitForPipelineExecution(SageMakerClient sageMakerClient,
String executionArn)
    throws InterruptedException {
    String status;
    int index = 0;
    do {
        DescribePipelineExecutionRequest pipelineExecutionRequest =
DescribePipelineExecutionRequest.builder()
            .pipelineExecutionArn(executionArn)
            .build();

        DescribePipelineExecutionResponse response = sageMakerClient
            .describePipelineExecution(pipelineExecutionRequest);
        status = response.pipelineExecutionStatusAsString();
        System.out.println(index + ". The Status of the pipeline is " + status);
        TimeUnit.SECONDS.sleep(4);
        index++;
    } while ("Executing".equals(status));
    System.out.println("Pipeline finished with status " + status);
}
```

- Per i dettagli sull'API, consulta la [DescribePipelineExecution](#) sezione AWS SDK for Java 2.x API Reference.

## StartPipelineExecution

Il seguente esempio di codice mostra come utilizzare `StartPipelineExecution`.

## SDK per Java 2.x

 Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
// Start a pipeline run with job configurations.
public static String executePipeline(SageMakerClient sageMakerClient, String
bucketName, String queueUrl,
    String roleArn, String pipelineName) {
    System.out.println("Starting pipeline execution.");
    String inputBucketLocation = "s3://" + bucketName + "/samplefiles/
latlongtest.csv";
    String output = "s3://" + bucketName + "/outputfiles/";
    Gson gson = new GsonBuilder()
        .setFieldNamingPolicy(FieldNamingPolicy.UPPER_CAMEL_CASE)
        .setPrettyPrinting().create();

    // Set up all parameters required to start the pipeline.
    List<Parameter> parameters = new ArrayList<>();
    Parameter para1 = Parameter.builder()
        .name("parameter_execution_role")
        .value(roleArn)
        .build();

    Parameter para2 = Parameter.builder()
        .name("parameter_queue_url")
        .value(queueUrl)
        .build();

    String inputJSON = "{\n" +
        "  \"DataSourceConfig\": {\n" +
        "    \"S3Data\": {\n" +
        "      \"S3Uri\": \"s3://" + bucketName + "/samplefiles/
latlongtest.csv\"\n" +
        "    },\n" +
        "    \"Type\": \"S3_DATA\"\n" +
        "  },\n" +
        "  \"DocumentType\": \"CSV\"\n" +
        "}";
```

```
System.out.println(inputJSON);

Parameter para3 = Parameter.builder()
    .name("parameter_vej_input_config")
    .value(inputJSON)
    .build();

// Create an ExportVectorEnrichmentJobOutputConfig object.
VectorEnrichmentJobS3Data jobS3Data = VectorEnrichmentJobS3Data.builder()
    .s3Uri(output)
    .build();

ExportVectorEnrichmentJobOutputConfig outputConfig =
ExportVectorEnrichmentJobOutputConfig.builder()
    .s3Data(jobS3Data)
    .build();

String gson4 = gson.toJson(outputConfig);
Parameter para4 = Parameter.builder()
    .name("parameter_vej_export_config")
    .value(gson4)
    .build();

System.out.println("parameter_vej_export_config:" +
gson.toJson(outputConfig));

// Create a VectorEnrichmentJobConfig object.
ReverseGeocodingConfig reverseGeocodingConfig =
ReverseGeocodingConfig.builder()
    .xAttributeName("Longitude")
    .yAttributeName("Latitude")
    .build();

VectorEnrichmentJobConfig jobConfig = VectorEnrichmentJobConfig.builder()
    .reverseGeocodingConfig(reverseGeocodingConfig)
    .build();

String para5JSON = "{\"MapMatchingConfig\":null,\"ReverseGeocodingConfig\":
{\"XAttributeName\":\"Longitude\",\"YAttributeName\":\"Latitude\"}}";
Parameter para5 = Parameter.builder()
    .name("parameter_step_1_vej_config")
    .value(para5JSON)
    .build();
```

```
System.out.println("parameter_step_1_vej_config:" + gson.toJson(jobConfig));
parameters.add(para1);
parameters.add(para2);
parameters.add(para3);
parameters.add(para4);
parameters.add(para5);

StartPipelineExecutionRequest pipelineExecutionRequest =
StartPipelineExecutionRequest.builder()
    .pipelineExecutionDescription("Created using Java SDK")
    .pipelineExecutionDisplayName(pipelineName + "-example-execution")
    .pipelineParameters(parameters)
    .pipelineName(pipelineName)
    .build();

StartPipelineExecutionResponse response =
sageMakerClient.startPipelineExecution(pipelineExecutionRequest);
return response.pipelineExecutionArn();
}
```

- Per i dettagli sull'API, consulta la [StartPipelineExecution](#) sezione AWS SDK for Java 2.x API Reference.

## Scenari

Inizia con i lavori e le pipeline geospaziali

L'esempio di codice seguente mostra come:

- Imposta le risorse per una pipeline.
- Configura una pipeline che esegua un lavoro geospaziale.
- Avvio dell'esecuzione di una pipeline.
- Monitora lo stato dell'esecuzione.
- Visualizza l'output della pipeline.
- Pulisci le risorse.

Per ulteriori informazioni, consulta [Creare ed eseguire SageMaker pipeline utilizzando gli AWS SDK su Community.aws](#).

## SDK per Java 2.x

 Note

C'è di più su. [GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.](#)

```
public class SagemakerWorkflow {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");
    private static String eventSourceMapping = "";

    public static void main(String[] args) throws InterruptedException {
        final String usage = "\n" +
            "Usage:\n" +
            "    <sageMakerRoleName> <lambdaRoleName> <functionFileLocation>
<functionName> <queueName> <bucketName> <lnglatData> <spatialPipelinePath>
<pipelineName>\n\n"
            +
            "Where:\n" +
            "    sageMakerRoleName - The name of the Amazon SageMaker role.\n\n"
+
            "    lambdaRoleName - The name of the AWS Lambda role.\n\n" +
            "    functionFileLocation - The file location where the JAR file
that represents the AWS Lambda function is located.\n\n"
            +
            "    functionName - The name of the AWS Lambda function (for
example, SageMakerExampleFunction).\n\n" +
            "    queueName - The name of the Amazon Simple Queue Service (Amazon
SQS) queue.\n\n" +
            "    bucketName - The name of the Amazon Simple Storage Service
(Amazon S3) bucket.\n\n" +
            "    lnglatData - The file location of the latlongtest.csv file
required for this use case.\n\n" +
            "    spatialPipelinePath - The file location of the
GeoSpatialPipeline.json file required for this use case.\n\n"
            +
            "    pipelineName - The name of the pipeline to create (for example,
sagemaker-sdk-example-pipeline).\n\n";

        if (args.length != 9) {
            System.out.println(usage);
        }
    }
}
```



```
        System.exit(1);
    }

    String sageMakerRoleName = args[0];
    String lambdaRoleName = args[1];
    String functionFileLocation = args[2];
    String functionName = args[3];
    String queueName = args[4];
    String bucketName = args[5];
    String lnglatData = args[6];
    String spatialPipelinePath = args[7];
    String pipelineName = args[8];
    String handlerName = "org.example.SageMakerLambdaFunction::handleRequest";

    Region region = Region.US_WEST_2;
    SageMakerClient sageMakerClient = SageMakerClient.builder()
        .region(region)
        .build();

    IamClient iam = IamClient.builder()
        .region(region)
        .build();

    LambdaClient lambdaClient = LambdaClient.builder()
        .region(region)
        .build();

    SqsClient sqsClient = SqsClient.builder()
        .region(region)
        .build();

    S3Client s3Client = S3Client.builder()
        .region(region)
        .build();

    System.out.println(DASHES);
    System.out.println("Welcome to the Amazon SageMaker pipeline example
scenario.");
    System.out.println(
        "\nThis example workflow will guide you through setting up and
running an" +
            "\nAmazon SageMaker pipeline. The pipeline uses an AWS
Lambda function and an" +
```

```
        "\nAmazon SQS Queue. It runs a vector enrichment reverse
geocode job to" +
        "\nreverse geocode addresses in an input file and store the
results in an export file.");
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("First, we will set up the roles, functions, and queue
needed by the SageMaker pipeline.");
    String lambdaRoleArn = checkLambdaRole(iam, lambdaRoleName);
    String sageMakerRoleArn = checkSageMakerRole(iam, sageMakerRoleName);

    String functionArn = checkFunction(lambdaClient, functionName,
functionFileLocation, lambdaRoleArn,
        handlerName);
    String queueUrl = checkQueue(sqsClient, lambdaClient, queueName,
functionName);
    System.out.println("The queue URL is " + queueUrl);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("Setting up bucket " + bucketName);
    if (!checkBucket(s3Client, bucketName)) {
        setupBucket(s3Client, bucketName);
        System.out.println("Put " + lnglatData + " into " + bucketName);
        putS3Object(s3Client, bucketName, "latlongtest.csv", lnglatData);
    }
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("Now we can create and run our pipeline.");
    setupPipeline(sageMakerClient, spatialPipelinePath, sageMakerRoleArn,
functionArn, pipelineName);
    String pipelineExecutionARN = executePipeline(sageMakerClient, bucketName,
queueUrl, sageMakerRoleArn,
        pipelineName);
    System.out.println("The pipeline execution ARN value is " +
pipelineExecutionARN);
    waitForPipelineExecution(sageMakerClient, pipelineExecutionARN);
    System.out.println("Getting output results " + bucketName);
    getOutputResults(s3Client, bucketName);
    System.out.println(DASHES);

    System.out.println(DASHES);
```

```

        System.out.println("The pipeline has completed. To view the pipeline and
runs " +
        "in SageMaker Studio, follow these instructions:" +
        "\nhttps://docs.aws.amazon.com/sagemaker/latest/dg/pipelines-
studio.html");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Do you want to delete the AWS resources used in this
Workflow? (y/n)");
        Scanner in = new Scanner(System.in);
        String delResources = in.nextLine();
        if (delResources.compareTo("y") == 0) {
            System.out.println("Lets clean up the AWS resources. Wait 30 seconds");
            TimeUnit.SECONDS.sleep(30);
            deleteEventSourceMapping(lambdaClient);
            deleteSQSQueue(sqsClient, queueName);
            listBucketObjects(s3Client, bucketName);
            deleteBucket(s3Client, bucketName);
            deleteLambdaFunction(lambdaClient, functionName);
            deleteLambdaRole(iam, lambdaRoleName);
            deleteSagemakerRole(iam, sageMakerRoleName);
            deletePipeline(sageMakerClient, pipelineName);
        } else {
            System.out.println("The AWS Resources were not deleted!");
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("SageMaker pipeline scenario is complete.");
        System.out.println(DASHES);
    }

    private static void readObject(S3Client s3Client, String bucketName, String key)
    {
        System.out.println("Output file contents: \n");
        GetObjectRequest objectRequest = GetObjectRequest.builder()
            .bucket(bucketName)
            .key(key)
            .build();

        ResponseBytes<GetObjectResponse> objectBytes =
s3Client.getObjectAsBytes(objectRequest);
        byte[] byteArray = objectBytes.asByteArray();
    }

```

```
String text = new String(byteArray, StandardCharsets.UTF_8);
System.out.println("Text output: " + text);
}

// Display some results from the output directory.
public static void getOutputResults(S3Client s3Client, String bucketName) {
    System.out.println("Getting output results {bucketName}.");
    ListObjectsRequest listObjectsRequest = ListObjectsRequest.builder()
        .bucket(bucketName)
        .prefix("outputfiles/")
        .build();

    ListObjectsResponse response = s3Client.listObjects(listObjectsRequest);
    List<S3Object> s3Objects = response.contents();
    for (S3Object object : s3Objects) {
        readObject(s3Client, bucketName, object.key());
    }
}

// Check the status of a pipeline execution.
public static void waitForPipelineExecution(SageMakerClient sageMakerClient,
String executionArn)
    throws InterruptedException {
    String status;
    int index = 0;
    do {
        DescribePipelineExecutionRequest pipelineExecutionRequest =
DescribePipelineExecutionRequest.builder()
            .pipelineExecutionArn(executionArn)
            .build();

        DescribePipelineExecutionResponse response = sageMakerClient
            .describePipelineExecution(pipelineExecutionRequest);
        status = response.pipelineExecutionStatusAsString();
        System.out.println(index + ". The Status of the pipeline is " + status);
        TimeUnit.SECONDS.sleep(4);
        index++;
    } while ("Executing".equals(status));
    System.out.println("Pipeline finished with status " + status);
}

// Delete a SageMaker pipeline by name.
public static void deletePipeline(SageMakerClient sageMakerClient, String
pipelineName) {
```

```
DeletePipelineRequest pipelineRequest = DeletePipelineRequest.builder()
    .pipelineName(pipelineName)
    .build();

sageMakerClient.deletePipeline(pipelineRequest);
System.out.println("*** Successfully deleted " + pipelineName);
}

// Create a pipeline from the example pipeline JSON.
public static void setupPipeline(SageMakerClient sageMakerClient, String
filePath, String roleArn,
    String functionArn, String pipelineName) {
    System.out.println("Setting up the pipeline.");
    JSONParser parser = new JSONParser();

    // Read JSON and get pipeline definition.
    try (FileReader reader = new FileReader(filePath)) {
        Object obj = parser.parse(reader);
        JSONObject jsonObject = (JSONObject) obj;
        JSONArray stepsArray = (JSONArray) jsonObject.get("Steps");
        for (Object stepObj : stepsArray) {
            JSONObject step = (JSONObject) stepObj;
            if (step.containsKey("FunctionArn")) {
                step.put("FunctionArn", functionArn);
            }
        }
    }
    System.out.println(jsonObject);

    // Create the pipeline.
    CreatePipelineRequest pipelineRequest = CreatePipelineRequest.builder()
        .pipelineDescription("Java SDK example pipeline")
        .roleArn(roleArn)
        .pipelineName(pipelineName)
        .pipelineDefinition(jsonObject.toString())
        .build();

    sageMakerClient.createPipeline(pipelineRequest);

} catch (IamException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
} catch (IOException | ParseException e) {
    throw new RuntimeException(e);
}
```

```

}

// Start a pipeline run with job configurations.
public static String executePipeline(SageMakerClient sageMakerClient, String
bucketName, String queueUrl,
    String roleArn, String pipelineName) {
    System.out.println("Starting pipeline execution.");
    String inputBucketLocation = "s3://" + bucketName + "/samplefiles/
latlongtest.csv";
    String output = "s3://" + bucketName + "/outputfiles/";
    Gson gson = new GsonBuilder()
        .setFieldNamingPolicy(FieldNamingPolicy.UPPER_CAMEL_CASE)
        .setPrettyPrinting().create();

    // Set up all parameters required to start the pipeline.
    List<Parameter> parameters = new ArrayList<>();
    Parameter para1 = Parameter.builder()
        .name("parameter_execution_role")
        .value(roleArn)
        .build();

    Parameter para2 = Parameter.builder()
        .name("parameter_queue_url")
        .value(queueUrl)
        .build();

    String inputJSON = "{\n" +
        "  \"DataSourceConfig\": {\n" +
        "    \"S3Data\": {\n" +
        "      \"S3Uri\": \"s3://" + bucketName + "/samplefiles/
latlongtest.csv\"\n" +
        "    },\n" +
        "    \"Type\": \"S3_DATA\"\n" +
        "  },\n" +
        "  \"DocumentType\": \"CSV\"\n" +
        "}";

    System.out.println(inputJSON);

    Parameter para3 = Parameter.builder()
        .name("parameter_vej_input_config")
        .value(inputJSON)
        .build();

```

```
// Create an ExportVectorEnrichmentJobOutputConfig object.
VectorEnrichmentJobS3Data jobS3Data = VectorEnrichmentJobS3Data.builder()
    .s3Uri(output)
    .build();

ExportVectorEnrichmentJobOutputConfig outputConfig =
ExportVectorEnrichmentJobOutputConfig.builder()
    .s3Data(jobS3Data)
    .build();

String gson4 = gson.toJson(outputConfig);
Parameter para4 = Parameter.builder()
    .name("parameter_vej_export_config")
    .value(gson4)
    .build();

System.out.println("parameter_vej_export_config:" +
gson.toJson(outputConfig));

// Create a VectorEnrichmentJobConfig object.
ReverseGeocodingConfig reverseGeocodingConfig =
ReverseGeocodingConfig.builder()
    .xAttributeName("Longitude")
    .yAttributeName("Latitude")
    .build();

VectorEnrichmentJobConfig jobConfig = VectorEnrichmentJobConfig.builder()
    .reverseGeocodingConfig(reverseGeocodingConfig)
    .build();

String para5JSON = "{\"MapMatchingConfig\":null,\"ReverseGeocodingConfig\":
{\"XAttributeName\":\"Longitude\",\"YAttributeName\":\"Latitude\"}}";
Parameter para5 = Parameter.builder()
    .name("parameter_step_1_vej_config")
    .value(para5JSON)
    .build();

System.out.println("parameter_step_1_vej_config:" + gson.toJson(jobConfig));
parameters.add(para1);
parameters.add(para2);
parameters.add(para3);
parameters.add(para4);
parameters.add(para5);
```

```
        StartPipelineExecutionRequest pipelineExecutionRequest =
StartPipelineExecutionRequest.builder()
    .pipelineExecutionDescription("Created using Java SDK")
    .pipelineExecutionDisplayName(pipelineName + "-example-execution")
    .pipelineParameters(parameters)
    .pipelineName(pipelineName)
    .build();

        StartPipelineExecutionResponse response =
sageMakerClient.startPipelineExecution(pipelineExecutionRequest);
        return response.pipelineExecutionArn();
    }

    public static void deleteEventSourceMapping(LambdaClient lambdaClient) {
        DeleteEventSourceMappingRequest eventSourceMappingRequest =
DeleteEventSourceMappingRequest.builder()
    .uuid(eventSourceMapping)
    .build();

        lambdaClient.deleteEventSourceMapping(eventSourceMappingRequest);
    }

    public static void deleteSagemakerRole(IamClient iam, String roleName) {
        String[] sageMakerRolePolicies = getSageMakerRolePolicies();
        try {
            for (String policy : sageMakerRolePolicies) {
                // First the policy needs to be detached.
                DetachRolePolicyRequest rolePolicyRequest =
DetachRolePolicyRequest.builder()
    .policyArn(policy)
    .roleName(roleName)
    .build();

                iam.detachRolePolicy(rolePolicyRequest);
            }

            // Delete the role.
            DeleteRoleRequest roleRequest = DeleteRoleRequest.builder()
    .roleName(roleName)
    .build();

            iam.deleteRole(roleRequest);
            System.out.println("*** Successfully deleted " + roleName);
        }
    }
}
```



```
    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void deleteLambdaRole(IamClient iam, String roleName) {
    String[] lambdaRolePolicies = getLambdaRolePolicies();
    try {
        for (String policy : lambdaRolePolicies) {
            // First the policy needs to be detached.
            DetachRolePolicyRequest rolePolicyRequest =
DetachRolePolicyRequest.builder()
                .policyArn(policy)
                .roleName(roleName)
                .build();

            iam.detachRolePolicy(rolePolicyRequest);
        }

        // Delete the role.
        DeleteRoleRequest roleRequest = DeleteRoleRequest.builder()
            .roleName(roleName)
            .build();

        iam.deleteRole(roleRequest);
        System.out.println("*** Successfully deleted " + roleName);

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

// Delete the specific AWS Lambda function.
public static void deleteLambdaFunction(LambdaClient awsLambda, String
functionName) {
    try {
        DeleteFunctionRequest request = DeleteFunctionRequest.builder()
            .functionName(functionName)
            .build();

        awsLambda.deleteFunction(request);
        System.out.println("*** " + functionName + " was deleted");
    }
}
```

```
    } catch (LambdaException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

// Delete the specific S3 bucket.
public static void deleteBucket(S3Client s3Client, String bucketName) {
    DeleteBucketRequest deleteBucketRequest = DeleteBucketRequest.builder()
        .bucket(bucketName)
        .build();
    s3Client.deleteBucket(deleteBucketRequest);
    System.out.println("*** " + bucketName + " was deleted.");
}

public static void listBucketObjects(S3Client s3, String bucketName) {
    try {
        ListObjectsRequest listObjects = ListObjectsRequest
            .builder()
            .bucket(bucketName)
            .build();

        ListObjectsResponse res = s3.listObjects(listObjects);
        List<S3Object> objects = res.contents();
        for (S3Object myValue : objects) {
            System.out.print("\n The name of the key is " + myValue.key());
            deleteBucketObjects(s3, bucketName, myValue.key());
        }

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void deleteBucketObjects(S3Client s3, String bucketName, String
objectName) {
    ArrayList<ObjectIdentifier> toDelete = new ArrayList<>();
    toDelete.add(ObjectIdentifier.builder()
        .key(objectName)
        .build());
    try {
        DeleteObjectsRequest dor = DeleteObjectsRequest.builder()
```

```
        .bucket(bucketName)
        .delete(Delete.builder()
            .objects(toDelete).build())
        .build();

s3.deleteObjects(dor);
System.out.println("*** " + bucketName + " objects were deleted.");

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

// Delete the specific Amazon SQS queue.
public static void deleteSQSQueue(SqsClient sqsClient, String queueName) {
    try {
        GetQueueUrlRequest getQueueRequest = GetQueueUrlRequest.builder()
            .queueName(queueName)
            .build();

        String queueUrl = sqsClient.getQueueUrl(getQueueRequest).queueUrl();
        DeleteQueueRequest deleteQueueRequest = DeleteQueueRequest.builder()
            .queueUrl(queueUrl)
            .build();

        sqsClient.deleteQueue(deleteQueueRequest);

    } catch (SqsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void putS3Object(S3Client s3, String bucketName, String objectKey,
String objectPath) {
    try {
        Map<String, String> metadata = new HashMap<>();
        metadata.put("x-amz-meta-myVal", "test");
        PutObjectRequest putOb = PutObjectRequest.builder()
            .bucket(bucketName)
            .key("samplefiles/" + objectKey)
            .metadata(metadata)
            .build();
    }
```

```
s3.putObject(putOb, RequestBody.fromFile(new File(objectPath)));
System.out.println("Successfully placed " + objectKey + " into bucket "
+ bucketName);

    } catch (S3Exception e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void setupBucket(S3Client s3Client, String bucketName) {
    try {
        S3Waiter s3Waiter = s3Client.waiter();
        CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
            .bucket(bucketName)
            .build();

        s3Client.createBucket(bucketRequest);
        HeadBucketRequest bucketRequestWait = HeadBucketRequest.builder()
            .bucket(bucketName)
            .build();

        // Wait until the bucket is created and print out the response.
        WaiterResponse<HeadBucketResponse> waiterResponse =
s3Waiter.waitUntilBucketExists(bucketRequestWait);
        waiterResponse.matched().response().ifPresent(System.out::println);
        System.out.println(bucketName + " is ready");

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

// Set up the SQS queue to use with the pipeline.
public static String setupQueue(SqsClient sqsClient, LambdaClient lambdaClient,
String queueName,
    String lambdaName) {
    System.out.println("Setting up queue named " + queueName);
    try {
        Map<QueueAttributeName, String> queueAtt = new HashMap<>();
        queueAtt.put(QueueAttributeName.DELAY_SECONDS, "5");
        queueAtt.put(QueueAttributeName.RECEIVE_MESSAGE_WAIT_TIME_SECONDS, "5");
```

```

        queueAtt.put(QueueAttributeName.VISIBILITY_TIMEOUT, "300");
        CreateQueueRequest createQueueRequest = CreateQueueRequest.builder()
            .queueName(queueName)
            .attributes(queueAtt)
            .build();

        sqsClient.createQueue(createQueueRequest);
        System.out.println("\nGet queue url");
        GetQueueUrlResponse getQueueUrlResponse = sqsClient

.getQueueUrl(GetQueueUrlRequest.builder().queueName(queueName).build());
        TimeUnit.SECONDS.sleep(15);

        connectLambda(sqsClient, lambdaClient, getQueueUrlResponse.queueUrl(),
lambdaName);
        System.out.println("Queue ready with Url " +
getQueueUrlResponse.queueUrl());
        return getQueueUrlResponse.queueUrl();

    } catch (SqsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    } catch (InterruptedException e) {
        throw new RuntimeException(e);
    }
    return "";
}

// Connect the queue to the Lambda function as an event source.
public static void connectLambda(SqsClient sqsClient, LambdaClient lambdaClient,
String queueUrl,
    String lambdaName) {
    System.out.println("Connecting the Lambda function and queue for the
pipeline.");
    String queueArn = "";

    // Specify the attributes to retrieve.
    List<QueueAttributeName> atts = new ArrayList<>();
    atts.add(QueueAttributeName.QUEUE_ARN);
    GetQueueAttributesRequest attributesRequest =
GetQueueAttributesRequest.builder()
        .queueUrl(queueUrl)
        .attributeNames(atts)
        .build();

```

```
    GetQueueAttributesResponse response =
sqscClient.getQueueAttributes(attributesRequest);
    Map<String, String> queueAtts = response.attributesAsStrings();
    for (Map.Entry<String, String> queueAtt : queueAtts.entrySet()) {
        System.out.println("Key = " + queueAtt.getKey() + ", Value = " +
queueAtt.getValue());
        queueArn = queueAtt.getValue();
    }

    CreateEventSourceMappingRequest eventSourceMappingRequest =
CreateEventSourceMappingRequest.builder()
        .eventSourceArn(queueArn)
        .functionName(lambdaName)
        .build();

    CreateEventSourceMappingResponse response1 =
lambdaClient.createEventSourceMapping(eventSourceMappingRequest);
    eventSourceMapping = response1.uuid();
    System.out.println("The mapping between the event source and Lambda function
was successful");
}

// Create an AWS Lambda function.
public static String createLambdaFunction(LambdaClient awsLambda, String
functionName, String filePath, String role,
String handler) {
    try {
        LambdaWaiter waiter = awsLambda.waiter();
        InputStream is = new FileInputStream(filePath);
        SdkBytes fileToUpload = SdkBytes.fromInputStream(is);
        FunctionCode code = FunctionCode.builder()
            .zipFile(fileToUpload)
            .build();

        CreateFunctionRequest functionRequest = CreateFunctionRequest.builder()
            .functionName(functionName)
            .description("SageMaker example function.")
            .code(code)
            .handler(handler)
            .runtime(Runtime.JAVA11)
            .timeout(200)
            .memorySize(1024)
            .role(role)
    }
```

```

        .build());

        // Create a Lambda function using a waiter.
        CreateFunctionResponse functionResponse =
awsLambda.createFunction(functionRequest);
        GetFunctionRequest getFunctionRequest = GetFunctionRequest.builder()
            .functionName(functionName)
            .build();
        WaiterResponse<GetFunctionResponse> waiterResponse =
waiter.waitUntilFunctionExists(getFunctionRequest);
        waiterResponse.matched().response().ifPresent(System.out::println);
        System.out.println("The function ARN is " +
functionResponse.functionArn());
        return functionResponse.functionArn();

    } catch (LambdaException | FileNotFoundException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return "";
}

public static String createSageMakerRole(IamClient iam, String roleName) {
    String[] sageMakerRolePolicies = getSageMakerRolePolicies();
    System.out.println("Creating a role to use with SageMaker.");
    String assumeRolePolicy = "{" +
        "\"Version\": \"2012-10-17\", " +
        "\"Statement\": [{" +
        "\"Effect\": \"Allow\", " +
        "\"Principal\": {" +
        "\"Service\": [" +
        "\"sagemaker.amazonaws.com\", " +
        "\"sagemaker-geospatial.amazonaws.com\", " +
        "\"lambda.amazonaws.com\", " +
        "\"s3.amazonaws.com\"" +
        "]" +
        "}, " +
        "\"Action\": \"sts:AssumeRole\"" +
        "}] " +
        "}";

    try {
        CreateRoleRequest request = CreateRoleRequest.builder()
            .roleName(roleName)

```

```

        .assumeRolePolicyDocument(assumeRolePolicy)
        .description("Created using the AWS SDK for Java")
        .build();

    CreateRoleResponse roleResult = iam.createRole(request);

    // Attach the policies to the role.
    for (String policy : sageMakerRolePolicies) {
        AttachRolePolicyRequest attachRequest =
AttachRolePolicyRequest.builder()
            .roleName(roleName)
            .policyArn(policy)
            .build();

        iam.attachRolePolicy(attachRequest);
    }

    // Allow time for the role to be ready.
    TimeUnit.SECONDS.sleep(15);
    System.out.println("Role ready with ARN " + roleResult.role().arn());
    return roleResult.role().arn();

} catch (IamException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
} catch (InterruptedException e) {
    throw new RuntimeException(e);
}
return "";
}

private static String createLambdaRole(IamClient iam, String roleName) {
    String[] lambdaRolePolicies = getLambdaRolePolicies();
    String assumeRolePolicy = "{" +
        "\"Version\": \"2012-10-17\"," +
        "\"Statement\": [{" +
        "\"Effect\": \"Allow\"," +
        "\"Principal\": {" +
        "\"Service\": [" +
        "\"sagemaker.amazonaws.com\"," +
        "\"sagemaker-geospatial.amazonaws.com\"," +
        "\"lambda.amazonaws.com\"," +
        "\"s3.amazonaws.com\"" +
        "]" +
    }

```



```
        "}," +
        "\"Action\": \"sts:AssumeRole\"\" +
        "]" +
        "};

    try {
        CreateRoleRequest request = CreateRoleRequest.builder()
            .roleName(roleName)
            .assumeRolePolicyDocument(assumeRolePolicy)
            .description("Created using the AWS SDK for Java")
            .build();

        CreateRoleResponse roleResult = iam.createRole(request);

        // Attach the policies to the role.
        for (String policy : lambdaRolePolicies) {
            AttachRolePolicyRequest attachRequest =
AttachRolePolicyRequest.builder()
                .roleName(roleName)
                .policyArn(policy)
                .build();

            iam.attachRolePolicy(attachRequest);
        }

        // Allow time for the role to be ready.
        TimeUnit.SECONDS.sleep(15);
        System.out.println("Role ready with ARN " + roleResult.role().arn());
        return roleResult.role().arn();

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());

    } catch (InterruptedException e) {
        throw new RuntimeException(e);
    }
    return "";
}

public static String checkFunction(LambdaClient lambdaClient, String
functionName, String filePath, String role,
    String handler) {
    System.out.println("Create an AWS Lambda function used in this workflow.");
    String functionArn;
```

```
    try {
        // Does this function already exist.
        GetFunctionRequest functionRequest = GetFunctionRequest.builder()
            .functionName(functionName)
            .build();

        GetFunctionResponse response =
lambdaClient.getFunction(functionRequest);
        functionArn = response.configuration().functionArn();

    } catch (LambdaException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        functionArn = createLambdaFunction(lambdaClient, functionName, filePath,
role, handler);
    }
    return functionArn;
}

// Check to see if the specific S3 bucket exists. If the S3 bucket exists, this
// method returns true.
public static boolean checkBucket(S3Client s3, String bucketName) {
    try {
        HeadBucketRequest headBucketRequest = HeadBucketRequest.builder()
            .bucket(bucketName)
            .build();

        s3.headBucket(headBucketRequest);
        System.out.println(bucketName + " exists");
        return true;

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }
    return false;
}

// Checks to see if the Amazon SQS queue exists. If not, this method creates a
// new queue
// and returns the ARN value.
public static String checkQueue(SqsClient sqsClient, LambdaClient lambdaClient,
String queueName,
    String lambdaName) {
    System.out.println("Creating a queue for this use case.");
    String queueUrl;
```

```
    try {
        GetQueueUrlRequest request = GetQueueUrlRequest.builder()
            .queueName(queueName)
            .build();

        GetQueueUrlResponse response = sqsClient.getQueueUrl(request);
        queueUrl = response.queueUrl();
        System.out.println(queueUrl);

    } catch (SqsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        queueUrl = setupQueue(sqsClient, lambdaClient, queueName, lambdaName);
    }
    return queueUrl;
}

// Checks to see if the Lambda role exists. If not, this method creates it.
public static String checkLambdaRole(IamClient iam, String roleName) {
    System.out.println("Creating a role to for AWS Lambda to use.");
    String roleArn;
    try {
        GetRoleRequest roleRequest = GetRoleRequest.builder()
            .roleName(roleName)
            .build();

        GetRoleResponse response = iam.getRole(roleRequest);
        roleArn = response.role().arn();
        System.out.println(roleArn);

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        roleArn = createLambdaRole(iam, roleName);
    }
    return roleArn;
}

// Checks to see if the SageMaker role exists. If not, this method creates it.
public static String checkSageMakerRole(IamClient iam, String roleName) {
    System.out.println("Creating a role to for AWS SageMaker to use.");
    String roleArn;
    try {
        GetRoleRequest roleRequest = GetRoleRequest.builder()
            .roleName(roleName)
            .build();
```

```

        GetRoleResponse response = iam.getRole(roleRequest);
        roleArn = response.role().arn();
        System.out.println(roleArn);

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        roleArn = createSageMakerRole(iam, roleName);
    }
    return roleArn;
}

private static String[] getSageMakerRolePolicies() {
    String[] sageMakerRolePolicies = new String[3];
    sageMakerRolePolicies[0] = "arn:aws:iam::aws:policy/AmazonSageMakerFullAccess";
    sageMakerRolePolicies[1] = "arn:aws:iam::aws:policy/" +
    "AmazonSageMakerGeospatialFullAccess";
    sageMakerRolePolicies[2] = "arn:aws:iam::aws:policy/AmazonSQSFullAccess";
    return sageMakerRolePolicies;
}

private static String[] getLambdaRolePolicies() {
    String[] lambdaRolePolicies = new String[5];
    lambdaRolePolicies[0] = "arn:aws:iam::aws:policy/AmazonSageMakerFullAccess";
    lambdaRolePolicies[1] = "arn:aws:iam::aws:policy/AmazonSQSFullAccess";
    lambdaRolePolicies[2] = "arn:aws:iam::aws:policy/service-role/" +
    "AmazonSageMakerGeospatialFullAccess";
    lambdaRolePolicies[3] = "arn:aws:iam::aws:policy/service-role/"
        + "AmazonSageMakerServiceCatalogProductsLambdaServiceRolePolicy";
    lambdaRolePolicies[4] = "arn:aws:iam::aws:policy/service-role/" +
    "AWSLambdaSQSQueueExecutionRole";
    return lambdaRolePolicies;
}
}

```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella Documentazione di riferimento delle API AWS SDK for Java 2.x .
  - [CreatePipeline](#)
  - [DeletePipeline](#)
  - [DescribePipelineExecution](#)

- [StartPipelineExecution](#)
- [UpdatePipeline](#)

## Esempi di Secrets Manager con SDK for Java 2.x

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando AWS SDK for Java 2.x with Secrets Manager.

Le operazioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le operazioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati e negli esempi tra servizi.

Scenari: esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio.

Ogni esempio include un collegamento a GitHub, dove è possibile trovare istruzioni su come configurare ed eseguire il codice nel contesto.

### Argomenti

- [Azioni](#)

### Azioni

#### **GetSecretValue**

Il seguente esempio di codice mostra come utilizzare `GetSecretValue`.

SDK per Java 2.x

#### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.secretsmanager.SecretsManagerClient;
import software.amazon.awssdk.services.secretsmanager.model.GetSecretValueRequest;
import software.amazon.awssdk.services.secretsmanager.model.GetSecretValueResponse;
```

```
import software.amazon.awssdk.services.secretsmanager.model.SecretsManagerException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * We recommend that you cache your secret values by using client-side caching.
 *
 * Caching secrets improves speed and reduces your costs. For more information,
 * see the following documentation topic:
 *
 * https://docs.aws.amazon.com/secretsmanager/latest/userguide/retrieving-secrets.html
 */
public class GetSecretValue {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <secretName>\s

                Where:
                secretName - The name of the secret (for example, tutorials/
MyFirstSecret).\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String secretName = args[0];
        Region region = Region.US_EAST_1;
        SecretsManagerClient secretsClient = SecretsManagerClient.builder()
                .region(region)
                .build();

        getValue(secretsClient, secretName);
        secretsClient.close();
    }
}
```

```
public static void getValue(SecretsManagerClient secretsClient, String
secretName) {
    try {
        GetSecretValueRequest valueRequest = GetSecretValueRequest.builder()
            .secretId(secretName)
            .build();

        GetSecretValueResponse valueResponse =
secretsClient.getSecretValue(valueRequest);
        String secret = valueResponse.secretString();
        System.out.println(secret);

    } catch (SecretsManagerException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Per i dettagli sull'API, consulta la [GetSecretValue](#) sezione AWS SDK for Java 2.x API Reference.

## Esempi di Amazon SES con SDK for Java 2.x

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando AWS SDK for Java 2.x con Amazon SES.

Le operazioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le operazioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati e negli esempi tra servizi.

Scenari: esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio.

Ogni esempio include un collegamento a GitHub, dove puoi trovare istruzioni su come configurare ed eseguire il codice nel contesto.

### Argomenti

- [Azioni](#)

## Azioni

### ListIdentities

Il seguente esempio di codice mostra come utilizzare `ListIdentities`.

SDK per Java 2.x

#### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ses.SesClient;
import software.amazon.awssdk.services.ses.model.ListIdentitiesResponse;
import software.amazon.awssdk.services.ses.model.SesException;
import java.io.IOException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListIdentities {

    public static void main(String[] args) throws IOException {
        Region region = Region.US_WEST_2;
        SesClient client = SesClient.builder()
            .region(region)
            .build();

        listSESIIdentities(client);
    }

    public static void listSESIIdentities(SesClient client) {
        try {
            ListIdentitiesResponse identitiesResponse = client.listIdentities();
        }
    }
}
```



```
        List<String> identities = identitiesResponse.identities();
        for (String identity : identities) {
            System.out.println("The identity is " + identity);
        }

    } catch (SesException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Per i dettagli sull'API, consulta la [ListIdentities](#) sezione AWS SDK for Java 2.x API Reference.

## ListTemplates

Il seguente esempio di codice mostra come utilizzare `ListTemplates`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sesv2.SesV2Client;
import software.amazon.awssdk.services.sesv2.model.ListEmailTemplatesRequest;
import software.amazon.awssdk.services.sesv2.model.ListEmailTemplatesResponse;
import software.amazon.awssdk.services.sesv2.model.SesV2Exception;

public class ListTemplates {

    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        SesV2Client sesv2Client = SesV2Client.builder()
            .region(region)
            .build();

        listAllTemplates(sesv2Client);
    }
}
```

```
}

public static void listAllTemplates(SesV2Client sesv2Client) {
    try {
        ListEmailTemplatesRequest templatesRequest =
ListEmailTemplatesRequest.builder()
        .pageSize(1)
        .build();

        ListEmailTemplatesResponse response =
sesv2Client.listEmailTemplates(templatesRequest);
        response.templatesMetadata()
            .forEach(template -> System.out.println("Template name: " +
template.templateName()));

    } catch (SesV2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Per i dettagli sull'API, consulta la [List Templates](#) sezione AWS SDK for Java 2.x API Reference.

## SendEmail

Il seguente esempio di codice mostra come utilizzare `SendEmail`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ses.SesClient;
import software.amazon.awssdk.services.ses.model.Content;
import software.amazon.awssdk.services.ses.model.Destination;
import software.amazon.awssdk.services.ses.model.Message;
```

```
import software.amazon.awssdk.services.ses.model.Body;
import software.amazon.awssdk.services.ses.model.SendEmailRequest;
import software.amazon.awssdk.services.ses.model.SesException;

import javax.mail.MessagingException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class SendMessageEmailRequest {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <sender> <recipient> <subject>\s

            Where:
                sender - An email address that represents the sender.\s
                recipient - An email address that represents the recipient.\s
                subject - The subject line.\s

            """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String sender = args[0];
        String recipient = args[1];
        String subject = args[2];

        Region region = Region.US_EAST_1;
        SesClient client = SesClient.builder()
            .region(region)
            .build();

        // The HTML body of the email.
        String bodyHTML = "<html>" + "<head></head>" + "<body>" + "<h1>Hello!</h1>"
            + "<p> See the list of customers.</p>" + "</body>" + "</html>";
    }
}
```

```
try {
    send(client, sender, recipient, subject, bodyHTML);
    client.close();
    System.out.println("Done");
} catch (MessagingException e) {
    e.printStackTrace();
}
}

public static void send(SesClient client,
    String sender,
    String recipient,
    String subject,
    String bodyHTML) throws MessagingException {

    Destination destination = Destination.builder()
        .toAddresses(recipient)
        .build();

    Content content = Content.builder()
        .data(bodyHTML)
        .build();

    Content sub = Content.builder()
        .data(subject)
        .build();

    Body body = Body.builder()
        .html(content)
        .build();

    Message msg = Message.builder()
        .subject(sub)
        .body(body)
        .build();

    SendEmailRequest emailRequest = SendEmailRequest.builder()
        .destination(destination)
        .message(msg)
        .source(sender)
        .build();
}
```

```
        try {
            System.out.println("Attempting to send an email through Amazon SES " +
"using the AWS SDK for Java...");
            client.sendEmail(emailRequest);

        } catch (SesException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ses.SesClient;
import javax.activation.DataHandler;
import javax.activation.DataSource;
import javax.mail.Message;
import javax.mail.MessagingException;
import javax.mail.Session;
import javax.mail.internet.AddressException;
import javax.mail.internet.InternetAddress;
import javax.mail.internet.MimeMessage;
import javax.mail.internet.MimeMultipart;
import javax.mail.internet.MimeBodyPart;
import javax.mail.util.ByteArrayDataSource;
import java.io.ByteArrayOutputStream;
import java.io.IOException;
import java.nio.ByteBuffer;
import java.nio.file.Files;
import java.util.Properties;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.services.ses.model.SendRawEmailRequest;
import software.amazon.awssdk.services.ses.model.RawMessage;
import software.amazon.awssdk.services.ses.model.SesException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
```

```
public class SendMessageAttachment {
    public static void main(String[] args) throws IOException {
        final String usage = ""

            Usage:
                <sender> <recipient> <subject> <fileLocation>\s

            Where:
                sender - An email address that represents the sender.\s
                recipient - An email address that represents the recipient.\s
                subject - The subject line.\s
                fileLocation - The location of a Microsoft Excel file to use as
an attachment (C:/AWS/customers.xls).\s
                """;

        if (args.length != 4) {
            System.out.println(usage);
            System.exit(1);
        }

        String sender = args[0];
        String recipient = args[1];
        String subject = args[2];
        String fileLocation = args[3];

        // The email body for recipients with non-HTML email clients.
        String bodyText = "Hello,\r\n" + "Please see the attached file for a list "
            + "of customers to contact.";

        // The HTML body of the email.
        String bodyHTML = "<html>" + "<head></head>" + "<body>" + "<h1>Hello!</h1>"
            + "<p>Please see the attached file for a " + "list of customers to
contact.</p>" + "</body>"
            + "</html>";

        Region region = Region.US_WEST_2;
        SesClient client = SesClient.builder()
            .region(region)
            .build();

        try {
            sendemailAttachment(client, sender, recipient, subject, bodyText,
bodyHTML, fileLocation);
            client.close();
        }
    }
}
```

```
        System.out.println("Done");

    } catch (IOException | MessagingException e) {
        e.printStackTrace();
    }
}

public static void sendemailAttachment(SesClient client,
    String sender,
    String recipient,
    String subject,
    String bodyText,
    String bodyHTML,
    String fileLocation) throws AddressException, MessagingException,
IOException {

    java.io.File theFile = new java.io.File(fileLocation);
    byte[] fileContent = Files.readAllBytes(theFile.toPath());

    Session session = Session.getDefaultInstance(new Properties());

    // Create a new MimeMessage object.
    MimeMessage message = new MimeMessage(session);

    // Add subject, from and to lines.
    message.setSubject(subject, "UTF-8");
    message.setFrom(new InternetAddress(sender));
    message.setRecipients(Message.RecipientType.TO,
InternetAddress.parse(recipient));

    // Create a multipart/alternative child container.
    MimeMultipart msgBody = new MimeMultipart("alternative");

    // Create a wrapper for the HTML and text parts.
    MimeBodyPart wrap = new MimeBodyPart();

    // Define the text part.
    MimeBodyPart textPart = new MimeBodyPart();
    textPart.setContent(bodyText, "text/plain; charset=UTF-8");

    // Define the HTML part.
    MimeBodyPart htmlPart = new MimeBodyPart();
    htmlPart.setContent(bodyHTML, "text/html; charset=UTF-8");
```

```
// Add the text and HTML parts to the child container.
msgBody.addBodyPart(textPart);
msgBody.addBodyPart(htmlPart);

// Add the child container to the wrapper object.
wrap.setContent(msgBody);

// Create a multipart/mixed parent container.
MimeMultipart msg = new MimeMultipart("mixed");

// Add the parent container to the message.
message.setContent(msg);
msg.addBodyPart(wrap);

// Define the attachment.
MimeBodyPart att = new MimeBodyPart();
DataSource fds = new ByteArrayDataSource(fileContent,
    "application/vnd.openxmlformats-
officedocument.spreadsheetml.sheet");
att.setDataHandler(new DataHandler(fds));

String reportName = "WorkReport.xls";
att.setFileName(reportName);

// Add the attachment to the message.
msg.addBodyPart(att);

try {
    System.out.println("Attempting to send an email through Amazon SES " +
"using the AWS SDK for Java...");

    ByteArrayOutputStream outputStream = new ByteArrayOutputStream();
    message.writeTo(outputStream);

    ByteBuffer buf = ByteBuffer.wrap(outputStream.toByteArray());

    byte[] arr = new byte[buf.remaining()];
    buf.get(arr);

    SdkBytes data = SdkBytes.fromByteArray(arr);
    RawMessage rawMessage = RawMessage.builder()
        .data(data)
        .build();
```



```
        SendRawEmailRequest rawEmailRequest = SendRawEmailRequest.builder()
            .rawMessage(rawMessage)
            .build();

        client.sendRawEmail(rawEmailRequest);

    } catch (SesException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    System.out.println("Email sent using SesClient with attachment");
}
}
```

- Per i dettagli sull'API, consulta la [SendEmail](#) sezione AWS SDK for Java 2.x API Reference.

## SendTemplatedEmail

Il seguente esempio di codice mostra come utilizzare `SendTemplatedEmail`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sesv2.model.Destination;
import software.amazon.awssdk.services.sesv2.model.EmailContent;
import software.amazon.awssdk.services.sesv2.model.SendEmailRequest;
import software.amazon.awssdk.services.sesv2.model.SesV2Exception;
import software.amazon.awssdk.services.sesv2.SesV2Client;
import software.amazon.awssdk.services.sesv2.model.Template;

/**
 * Before running this AWS SDK for Java (v2) example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
```

```
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*
* Also, make sure that you create a template. See the following documentation
* topic:
*
* https://docs.aws.amazon.com/ses/latest/dg/send-personalized-email-api.html
*/

public class SendEmailTemplate {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <template> <sender> <recipient>\s

            Where:
                template - The name of the email template.
                sender - An email address that represents the sender.\s
                recipient - An email address that represents the recipient.\s
            """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String templateName = args[0];
        String sender = args[1];
        String recipient = args[2];
        Region region = Region.US_EAST_1;
        SesV2Client sesv2Client = SesV2Client.builder()
            .region(region)
            .build();

        send(sesv2Client, sender, recipient, templateName);
    }

    public static void send(SesV2Client client, String sender, String recipient,
String templateName) {
        Destination destination = Destination.builder()
            .toAddresses(recipient)
            .build();
    }
}
```

```
    /*
     * Specify both name and favorite animal (favoriteanimal) in your code when
     * defining the Template object.
     * If you don't specify all the variables in the template, Amazon SES
doesn't
     * send the email.
     */
    Template myTemplate = Template.builder()
        .templateName(templateName)
        .templateData("{\n" +
            "  \"name\": \"Jason\"\n," +
            "  \"favoriteanimal\": \"Cat\"\n" +
            "}")
        .build();

    EmailContent emailContent = EmailContent.builder()
        .template(myTemplate)
        .build();

    SendEmailRequest emailRequest = SendEmailRequest.builder()
        .destination(destination)
        .content(emailContent)
        .fromEmailAddress(sender)
        .build();

    try {
        System.out.println("Attempting to send an email based on a template
using the AWS SDK for Java (v2)...");
        client.sendEmail(emailRequest);
        System.out.println("email based on a template was sent");

    } catch (SesV2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Per i dettagli sull'API, consulta la [SendTemplatedEmail](#) sezione AWS SDK for Java 2.x API Reference.

## Esempi di API Amazon SES v2 con SDK for Java 2.x

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando l'API AWS SDK for Java 2.x with Amazon SES v2.

Le operazioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le operazioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati e negli esempi tra servizi.

Scenari: esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio.

Ogni esempio include un collegamento a GitHub, dove puoi trovare istruzioni su come configurare ed eseguire il codice nel contesto.

Argomenti

- [Azioni](#)
- [Scenari](#)

### Azioni

#### **CreateContact**

Il seguente esempio di codice mostra come utilizzare `CreateContact`.

SDK per Java 2.x

#### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
try {
    // Create a new contact with the provided email address in the
    CreateContactRequest contactRequest = CreateContactRequest.builder()
        .contactListName(CONTACT_LIST_NAME)
        .emailAddress(emailAddress)
        .build();
```

```
sesClient.createContact(contactRequest);
contacts.add(emailAddress);

System.out.println("Contact created: " + emailAddress);

// Send a welcome email to the new contact
String welcomeHtml = Files.readString(Paths.get("resources/
coupon_newsletter/welcome.html"));
String welcomeText = Files.readString(Paths.get("resources/
coupon_newsletter/welcome.txt"));

SendEmailRequest welcomeEmailRequest = SendEmailRequest.builder()
    .fromEmailAddress(this.verifiedEmail)
    .destination(Destination.builder().toAddresses(emailAddress).build())
    .content(EmailContent.builder()
        .simple(
            Message.builder()
                .subject(Content.builder().data("Welcome to the Weekly
Coupons Newsletter").build())
                .body(Body.builder()
                    .text(Content.builder().data(welcomeText).build())
                    .html(Content.builder().data(welcomeHtml).build())
                    .build())
                .build()
            )
        .build()
    )
    .build();
SendEmailResponse welcomeEmailResponse =
sesClient.sendEmail(welcomeEmailRequest);
System.out.println("Welcome email sent: " +
welcomeEmailResponse.messageId());
} catch (AlreadyExistsException e) {
    // If the contact already exists, skip this step for that contact and
    proceed
    // with the next contact
    System.out.println("Contact already exists, skipping creation...");
} catch (Exception e) {
    System.err.println("Error occurred while processing email address " +
emailAddress + ": " + e.getMessage());
    throw e;
}
}
```

- Per i dettagli sull'API, consulta la [CreateContact](#) sezione AWS SDK for Java 2.x API Reference.

## CreateContactList

Il seguente esempio di codice mostra come utilizzare `CreateContactList`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
try {
    // 2. Create a contact list
    String contactListName = CONTACT_LIST_NAME;
    CreateContactListRequest createContactListRequest =
CreateContactListRequest.builder()
        .contactListName(contactListName)
        .build();
    sesClient.createContactList(createContactListRequest);
    System.out.println("Contact list created: " + contactListName);
} catch (AlreadyExistsException e) {
    System.out.println("Contact list already exists, skipping creation: weekly-
coupons-newsletter");
} catch (LimitExceededException e) {
    System.err.println("Limit for contact lists has been exceeded.");
    throw e;
} catch (SesV2Exception e) {
    System.err.println("Error creating contact list: " + e.getMessage());
    throw e;
}
```

- Per i dettagli sull'API, consulta la [CreateContactList](#) sezione AWS SDK for Java 2.x API Reference.

## CreateEmailIdentity

Il seguente esempio di codice mostra come utilizzare `CreateEmailIdentity`.

## SDK per Java 2.x

 Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
try {
    CreateEmailIdentityRequest createEmailIdentityRequest =
CreateEmailIdentityRequest.builder()
    .emailIdentity(verifiedEmail)
    .build();
    sesClient.createEmailIdentity(createEmailIdentityRequest);
    System.out.println("Email identity created: " + verifiedEmail);
} catch (AlreadyExistsException e) {
    System.out.println("Email identity already exists, skipping creation: " +
verifiedEmail);
} catch (NotFoundException e) {
    System.err.println("The provided email address is not verified: " +
verifiedEmail);
    throw e;
} catch (LimitExceededException e) {
    System.err
        .println("You have reached the limit for email identities. Please remove
some identities and try again.");
    throw e;
} catch (SesV2Exception e) {
    System.err.println("Error creating email identity: " + e.getMessage());
    throw e;
}
```

- Per i dettagli sull'API, consulta la [CreateEmailIdentity](#) sezione AWS SDK for Java 2.x API Reference.

## CreateEmailTemplate

Il seguente esempio di codice mostra come utilizzare `CreateEmailTemplate`.

## SDK per Java 2.x

 Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
try {
    // Create an email template named "weekly-coupons"
    String newsletterHtml = loadFile("resources/coupon_newsletter/coupon-
newsletter.html");
    String newsletterText = loadFile("resources/coupon_newsletter/coupon-
newsletter.txt");

    CreateEmailTemplateRequest templateRequest =
CreateEmailTemplateRequest.builder()
    .templateName(TEMPLATE_NAME)
    .templateContent(EmailTemplateContent.builder()
        .subject("Weekly Coupons Newsletter")
        .html(newsletterHtml)
        .text(newsletterText)
        .build())
    .build();

    sesClient.createEmailTemplate(templateRequest);

    System.out.println("Email template created: " + TEMPLATE_NAME);
} catch (AlreadyExistsException e) {
    // If the template already exists, skip this step and proceed with the next
    // operation
    System.out.println("Email template already exists, skipping creation...");
} catch (LimitExceededException e) {
    // If the limit for email templates is exceeded, fail the workflow and inform
    // the user
    System.err.println("You have reached the limit for email templates. Please
remove some templates and try again.");
    throw e;
} catch (Exception e) {
    System.err.println("Error occurred while creating email template: " +
e.getMessage());
    throw e;
}
```



```
}
```

- Per i dettagli sull'API, consulta la [CreateEmailTemplate](#) sezione AWS SDK for Java 2.x API Reference.

## DeleteContactList

Il seguente esempio di codice mostra come utilizzare `DeleteContactList`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
try {
    // Delete the contact list
    DeleteContactListRequest deleteContactListRequest =
DeleteContactListRequest.builder()
    .contactListName(CONTACT_LIST_NAME)
    .build();

    sesClient.deleteContactList(deleteContactListRequest);

    System.out.println("Contact list deleted: " + CONTACT_LIST_NAME);
} catch (NotFoundException e) {
    // If the contact list does not exist, log the error and proceed
    System.out.println("Contact list not found. Skipping deletion...");
} catch (Exception e) {
    System.err.println("Error occurred while deleting the contact list: " +
e.getMessage());
    e.printStackTrace();
}
```

- Per i dettagli sull'API, consulta la [DeleteContactList](#) sezione AWS SDK for Java 2.x API Reference.

## DeleteEmailIdentity

Il seguente esempio di codice mostra come utilizzare `DeleteEmailIdentity`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
try {
    // Delete the email identity
    DeleteEmailIdentityRequest deleteIdentityRequest =
DeleteEmailIdentityRequest.builder()
    .emailIdentity(this.verifiedEmail)
    .build();

    sesClient.deleteEmailIdentity(deleteIdentityRequest);

    System.out.println("Email identity deleted: " + this.verifiedEmail);
} catch (NotFoundException e) {
    // If the email identity does not exist, log the error and proceed
    System.out.println("Email identity not found. Skipping deletion...");
} catch (Exception e) {
    System.err.println("Error occurred while deleting the email identity: " +
e.getMessage());
    e.printStackTrace();
}
} else {
    System.out.println("Skipping email identity deletion.");
}
```

- Per i dettagli sull'API, consulta la [DeleteEmailIdentity](#) sezione AWS SDK for Java 2.x API Reference.

## DeleteEmailTemplate

Il seguente esempio di codice mostra come utilizzare `DeleteEmailTemplate`.

## SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
try {
    // Delete the template
    DeleteEmailTemplateRequest deleteTemplateRequest =
DeleteEmailTemplateRequest.builder()
    .templateName(TEMPLATE_NAME)
    .build();

    sesClient.deleteEmailTemplate(deleteTemplateRequest);

    System.out.println("Email template deleted: " + TEMPLATE_NAME);
} catch (NotFoundException e) {
    // If the email template does not exist, log the error and proceed
    System.out.println("Email template not found. Skipping deletion...");
} catch (Exception e) {
    System.err.println("Error occurred while deleting the email template: " +
e.getMessage());
    e.printStackTrace();
}
```

- Per i dettagli sull'API, consulta la [DeleteEmailTemplate](#) sezione AWS SDK for Java 2.x API Reference.

## ListContacts

Il seguente esempio di codice mostra come utilizzare `ListContacts`.

## SDK per Java 2.x

 Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
ListContactsRequest contactListRequest = ListContactsRequest.builder()
    .contactListName(CONTACT_LIST_NAME)
    .build();

List<String> contactEmails;
try {
    ListContactsResponse contactListResponse =
sesClient.listContacts(contactListRequest);


    contactEmails = contactListResponse.contacts().stream()
        .map(Contact::emailAddress)
        .toList();
} catch (Exception e) {
    // TODO: Remove when listContacts's GET body issue is resolved.
    contactEmails = this.contacts;
}
```

- Per i dettagli sull'API, consulta la [ListContacts](#) sezione AWS SDK for Java 2.x API Reference.

## SendEmail

Il seguente esempio di codice mostra come utilizzare `SendEmail`.

## SDK per Java 2.x

 Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

## Invio di un messaggio

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sesv2.model.Body;
import software.amazon.awssdk.services.sesv2.model.Content;
import software.amazon.awssdk.services.sesv2.model.Destination;
import software.amazon.awssdk.services.sesv2.model.EmailContent;
import software.amazon.awssdk.services.sesv2.model.Message;
import software.amazon.awssdk.services.sesv2.model.SendEmailRequest;
import software.amazon.awssdk.services.sesv2.model.SesV2Exception;
import software.amazon.awssdk.services.sesv2.SesV2Client;

/**
 * Before running this AWS SDK for Java (v2) example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class SendEmail {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <sender> <recipient> <subject>\s

            Where:
                sender - An email address that represents the
sender.\s
                recipient - An email address that represents the
recipient.\s
                subject - The subject line.\s
            """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String sender = args[0];
        String recipient = args[1];
        String subject = args[2];
```

```
Region region = Region.US_EAST_1;
SesV2Client sesv2Client = SesV2Client.builder()
    .region(region)
    .build();

// The HTML body of the email.
String bodyHTML = "<html>" + "<head></head>" + "<body>" +
"<h1>Hello!</h1>"
    + "<p> See the list of customers.</p>" + "</body>" +
"</html>";

    send(sesv2Client, sender, recipient, subject, bodyHTML);
}

public static void send(SesV2Client client,
    String sender,
    String recipient,
    String subject,
    String bodyHTML) {

    Destination destination = Destination.builder()
        .toAddresses(recipient)
        .build();

    Content content = Content.builder()
        .data(bodyHTML)
        .build();

    Content sub = Content.builder()
        .data(subject)
        .build();

    Body body = Body.builder()
        .html(content)
        .build();

    Message msg = Message.builder()
        .subject(sub)
        .body(body)
        .build();

    EmailContent emailContent = EmailContent.builder()
        .simple(msg)
```

```

        .build();

        SendEmailRequest emailRequest = SendEmailRequest.builder()
            .destination(destination)
            .content(emailContent)
            .fromEmailAddress(sender)
            .build();

        try {
            System.out.println("Attempting to send an email through
Amazon SES "
                + "using the AWS SDK for Java...");
            client.sendEmail(emailRequest);
            System.out.println("email was sent");
        } catch (SesV2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}

```

Invia un messaggio utilizzando un modello.

```

String coupons = Files.readString(Paths.get("resources/coupon_newsletter/
sample_coupons.json"));
for (String emailAddress : contactEmails) {
    SendEmailRequest newsletterRequest = SendEmailRequest.builder()
        .destination(Destination.builder().toAddresses(emailAddress).build())
        .content(EmailContent.builder()
            .template(Template.builder()
                .templateName(TEMPLATE_NAME)
                .templateData(coupons)
                .build())
            .build())
        .fromEmailAddress(this.verifiedEmail)
        .listManagementOptions(ListManagementOptions.builder()
            .contactListName(CONTACT_LIST_NAME)
            .build())
        .build();
    SendEmailResponse newsletterResponse =
sesClient.sendEmail(newsletterRequest);
}
}

```

```
        System.out.println("Newsletter sent to " + emailAddress + ": " +
newsletterResponse.messageId());
    }
```

- Per i dettagli sull'API, consulta la [SendEmail](#) sezione AWS SDK for Java 2.x API Reference.

## Scenari

### Workflow della newsletter

Il seguente esempio di codice mostra come utilizzare il flusso di lavoro delle newsletter di Amazon SES API v2.

### SDK per Java 2.x

#### Note

C'è di più su [GitHub](#) Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
try {
    // 2. Create a contact list
    String contactListName = CONTACT_LIST_NAME;
    CreateContactListRequest createContactListRequest =
CreateContactListRequest.builder()
        .contactListName(contactListName)
        .build();
    sesClient.createContactList(createContactListRequest);
    System.out.println("Contact list created: " + contactListName);
} catch (AlreadyExistsException e) {
    System.out.println("Contact list already exists, skipping creation: weekly-
coupons-newsletter");
} catch (LimitExceededException e) {
    System.err.println("Limit for contact lists has been exceeded.");
    throw e;
} catch (SesV2Exception e) {
    System.err.println("Error creating contact list: " + e.getMessage());
    throw e;
}
```



```
try {
    // Create a new contact with the provided email address in the
    CreateContactRequest contactRequest = CreateContactRequest.builder()
        .contactListName(CONTACT_LIST_NAME)
        .emailAddress(emailAddress)
        .build();

    sesClient.createContact(contactRequest);
    contacts.add(emailAddress);

    System.out.println("Contact created: " + emailAddress);

    // Send a welcome email to the new contact
    String welcomeHtml = Files.readString(Paths.get("resources/
coupon_newsletter/welcome.html"));
    String welcomeText = Files.readString(Paths.get("resources/
coupon_newsletter/welcome.txt"));

    SendEmailRequest welcomeEmailRequest = SendEmailRequest.builder()
        .fromEmailAddress(this.verifiedEmail)
        .destination(Destination.builder().toAddresses(emailAddress).build())
        .content(EmailContent.builder()
            .simple(
                Message.builder()
                    .subject(Content.builder().data("Welcome to the Weekly
Coupons Newsletter").build())
                    .body(Body.builder()
                        .text(Content.builder().data(welcomeText).build())
                        .html(Content.builder().data(welcomeHtml).build())
                        .build())
                    .build())
            .build())
        .build();
    SendEmailResponse welcomeEmailResponse =
sesClient.sendEmail(welcomeEmailRequest);
    System.out.println("Welcome email sent: " +
welcomeEmailResponse.messageId());
} catch (AlreadyExistsException e) {
    // If the contact already exists, skip this step for that contact and
proceed
    // with the next contact
    System.out.println("Contact already exists, skipping creation...");
} catch (Exception e) {
```

```
        System.err.println("Error occurred while processing email address " +
emailAddress + ": " + e.getMessage());
        throw e;
    }
}

ListContactsRequest contactListRequest = ListContactsRequest.builder()
    .contactListName(CONTACT_LIST_NAME)
    .build();

List<String> contactEmails;
try {
    ListContactsResponse contactListResponse =
sesClient.listContacts(contactListRequest);

    contactEmails = contactListResponse.contacts().stream()
        .map(Contact::emailAddress)
        .toList();
} catch (Exception e) {
    // TODO: Remove when listContacts's GET body issue is resolved.
    contactEmails = this.contacts;
}

String coupons = Files.readString(Paths.get("resources/coupon_newsletter/
sample_coupons.json"));
for (String emailAddress : contactEmails) {
    SendEmailRequest newsletterRequest = SendEmailRequest.builder()
        .destination(Destination.builder().toAddresses(emailAddress).build())
        .content(EmailContent.builder()
            .template(Template.builder()
                .templateName(TEMPLATE_NAME)
                .templateData(coupons)
                .build())
            .build())
        .fromEmailAddress(this.verifiedEmail)
        .listManagementOptions(ListManagementOptions.builder()
            .contactListName(CONTACT_LIST_NAME)
            .build())
        .build();
    SendEmailResponse newsletterResponse =
sesClient.sendEmail(newsletterRequest);
    System.out.println("Newsletter sent to " + emailAddress + ": " +
newsletterResponse.messageId());
```

```
    }

    try {
        CreateEmailIdentityRequest createEmailIdentityRequest =
CreateEmailIdentityRequest.builder()
            .emailIdentity(verifiedEmail)
            .build();
        sesClient.createEmailIdentity(createEmailIdentityRequest);
        System.out.println("Email identity created: " + verifiedEmail);
    } catch (AlreadyExistsException e) {
        System.out.println("Email identity already exists, skipping creation: " +
verifiedEmail);
    } catch (NotFoundException e) {
        System.err.println("The provided email address is not verified: " +
verifiedEmail);
        throw e;
    } catch (LimitExceededException e) {
        System.err
            .println("You have reached the limit for email identities. Please remove
some identities and try again.");
        throw e;
    } catch (SesV2Exception e) {
        System.err.println("Error creating email identity: " + e.getMessage());
        throw e;
    }

    try {
        // Create an email template named "weekly-coupons"
        String newsletterHtml = loadFile("resources/coupon_newsletter/coupon-
newsletter.html");
        String newsletterText = loadFile("resources/coupon_newsletter/coupon-
newsletter.txt");

        CreateEmailTemplateRequest templateRequest =
CreateEmailTemplateRequest.builder()
            .templateName(TEMPLATE_NAME)
            .templateContent(EmailTemplateContent.builder()
                .subject("Weekly Coupons Newsletter")
                .html(newsletterHtml)
                .text(newsletterText)
                .build())
            .build();

        sesClient.createEmailTemplate(templateRequest);
    }
```

```
        System.out.println("Email template created: " + TEMPLATE_NAME);
    } catch (AlreadyExistsException e) {
        // If the template already exists, skip this step and proceed with the next
        // operation
        System.out.println("Email template already exists, skipping creation...");
    } catch (LimitExceededException e) {
        // If the limit for email templates is exceeded, fail the workflow and inform
        // the user
        System.err.println("You have reached the limit for email templates. Please
remove some templates and try again.");
        throw e;
    } catch (Exception e) {
        System.err.println("Error occurred while creating email template: " +
e.getMessage());
        throw e;
    }

    try {
        // Delete the contact list
        DeleteContactListRequest deleteContactListRequest =
DeleteContactListRequest.builder()
            .contactListName(CONTACT_LIST_NAME)
            .build();

        sesClient.deleteContactList(deleteContactListRequest);

        System.out.println("Contact list deleted: " + CONTACT_LIST_NAME);
    } catch (NotFoundException e) {
        // If the contact list does not exist, log the error and proceed
        System.out.println("Contact list not found. Skipping deletion...");
    } catch (Exception e) {
        System.err.println("Error occurred while deleting the contact list: " +
e.getMessage());
        e.printStackTrace();
    }

    try {
        // Delete the email identity
        DeleteEmailIdentityRequest deleteIdentityRequest =
DeleteEmailIdentityRequest.builder()
            .emailIdentity(this.verifiedEmail)
            .build();
```

```
sesClient.deleteEmailIdentity(deleteIdentityRequest);

System.out.println("Email identity deleted: " + this.verifiedEmail);
} catch (NotFoundException e) {
    // If the email identity does not exist, log the error and proceed
    System.out.println("Email identity not found. Skipping deletion...");
} catch (Exception e) {
    System.err.println("Error occurred while deleting the email identity: " +
e.getMessage());
    e.printStackTrace();
}
} else {
    System.out.println("Skipping email identity deletion.");
}

try {
    // Delete the template
    DeleteEmailTemplateRequest deleteTemplateRequest =
DeleteEmailTemplateRequest.builder()
    .templateName(TEMPLATE_NAME)
    .build();

    sesClient.deleteEmailTemplate(deleteTemplateRequest);

    System.out.println("Email template deleted: " + TEMPLATE_NAME);
} catch (NotFoundException e) {
    // If the email template does not exist, log the error and proceed
    System.out.println("Email template not found. Skipping deletion...");
} catch (Exception e) {
    System.err.println("Error occurred while deleting the email template: " +
e.getMessage());
    e.printStackTrace();
}
```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella Documentazione di riferimento delle API AWS SDK for Java 2.x .
  - [CreateContact](#)
  - [CreateContactList](#)
  - [CreateEmailIdentity](#)
  - [CreateEmailTemplate](#)

- [DeleteContactList](#)
- [DeleteEmailIdentity](#)
- [DeleteEmailTemplate](#)
- [ListContacts](#)
- [SendEmail.semplice](#)
- [SendEmail.modello](#)

## Esempi di Amazon SNS con SDK for Java 2.x

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando AWS SDK for Java 2.x con Amazon SNS.

Le operazioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le operazioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati e negli esempi tra servizi.

Scenari: esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio.

Ogni esempio include un collegamento a GitHub, dove puoi trovare istruzioni su come configurare ed eseguire il codice nel contesto.

Nozioni di base

Hello Amazon SNS

Gli esempi di codice seguenti mostrano come iniziare a utilizzare Amazon SNS.

SDK per Java 2.x

### Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
package com.example.sns;
```

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.paginators.ListTopicsIterable;

public class HelloSNS {
    public static void main(String[] args) {
        SnsClient snsClient = SnsClient.builder()
            .region(Region.US_EAST_1)
            .build();

        listSNSTopics(snsClient);
        snsClient.close();
    }

    public static void listSNSTopics(SnsClient snsClient) {
        try {
            ListTopicsIterable listTopics = snsClient.listTopicsPaginator();
            listTopics.stream()
                .flatMap(r -> r.topics().stream())
                .forEach(content -> System.out.println(" Topic ARN: " +
content.topicArn()));

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Per i dettagli sull'API, consulta la [ListTopics](#) sezione AWS SDK for Java 2.x API Reference.

## Argomenti

- [Azioni](#)
- [Scenari](#)
- [Esempi serverless](#)

## Azioni

### CheckIfPhoneNumberIsOptedOut

Il seguente esempio di codice mostra come utilizzare `CheckIfPhoneNumberIsOptedOut`.

SDK per Java 2.x

#### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import
    software.amazon.awssdk.services.sns.model.CheckIfPhoneNumberIsOptedOutRequest;
import
    software.amazon.awssdk.services.sns.model.CheckIfPhoneNumberIsOptedOutResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CheckOptOut {
    public static void main(String[] args) {

        final String usage = ""

            Usage:    <phoneNumber>

            Where:
                phoneNumber - The mobile phone number to look up (for example,
+1XXX5550100).

            """;
```



```
        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String phoneNumber = args[0];
        SnsClient snsClient = SnsClient.builder()
            .region(Region.US_EAST_1)
            .build();

        checkPhone(snsClient, phoneNumber);
        snsClient.close();
    }

    public static void checkPhone(SnsClient snsClient, String phoneNumber) {
        try {
            CheckIfPhoneNumberIsOptedOutRequest request =
                CheckIfPhoneNumberIsOptedOutRequest.builder()
                    .phoneNumber(phoneNumber)
                    .build();

            CheckIfPhoneNumberIsOptedOutResponse result =
                snsClient.checkIfPhoneNumberIsOptedOut(request);
            System.out.println(
                result.isOptedOut() + "Phone Number " + phoneNumber + " has
                Opted Out of receiving sns messages." +
                "\n\nStatus was " +
                result.sdkHttpResponse().statusCode());


        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Per i dettagli sull'API, consulta la [CheckIfPhoneNumberIsOptedOut](#) sezione AWS SDK for Java 2.x API Reference.

## ConfirmSubscription

Il seguente esempio di codice mostra come utilizzare `ConfirmSubscription`.

## SDK per Java 2.x

 Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.ConfirmSubscriptionRequest;
import software.amazon.awssdk.services.sns.model.ConfirmSubscriptionResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ConfirmSubscription {
    public static void main(String[] args) {
        final String usage = ""

                Usage:    <subscriptionToken> <topicArn>

                Where:
                    subscriptionToken - A short-lived token sent to an endpoint
during the Subscribe action.
                    topicArn - The ARN of the topic.\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String subscriptionToken = args[0];
        String topicArn = args[1];
        SnsClient snsClient = SnsClient.builder()
```

```
        .region(Region.US_EAST_1)
        .build();

    confirmSub(snsClient, subscriptionToken, topicArn);
    snsClient.close();
}

public static void confirmSub(SnsClient snsClient, String subscriptionToken,
String topicArn) {
    try {
        ConfirmSubscriptionRequest request =
ConfirmSubscriptionRequest.builder()
            .token(subscriptionToken)
            .topicArn(topicArn)
            .build();

        ConfirmSubscriptionResponse result =
snsClient.confirmSubscription(request);
        System.out.println("\n\nStatus was " +
result.sdkHttpResponse().statusCode() + "\n\nSubscription Arn: \n\n"
            + result.subscriptionArn());

    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Per i dettagli sull'API, consulta la [ConfirmSubscription](#) sezione AWS SDK for Java 2.x API Reference.

## CreateTopic

Il seguente esempio di codice mostra come utilizzare `CreateTopic`.

## SDK per Java 2.x

 Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.CreateTopicRequest;
import software.amazon.awssdk.services.sns.model.CreateTopicResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateTopic {
    public static void main(String[] args) {
        final String usage = ""

            Usage:    <topicName>

            Where:
                topicName - The name of the topic to create (for example,
mytopic).

            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String topicName = args[0];
        System.out.println("Creating a topic with name: " + topicName);
        SnsClient snsClient = SnsClient.builder()
```

```
        .region(Region.US_EAST_1)
        .build();

    String arnVal = createSNSTopic(snsClient, topicName);
    System.out.println("The topic ARN is" + arnVal);
    snsClient.close();
}

public static String createSNSTopic(SnsClient snsClient, String topicName) {
    CreateTopicResponse result;
    try {
        CreateTopicRequest request = CreateTopicRequest.builder()
            .name(topicName)
            .build();

        result = snsClient.createTopic(request);
        return result.topicArn();

    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
}
```

- Per i dettagli sull'API, consulta la [CreateTopic](#) sezione AWS SDK for Java 2.x API Reference.

## DeleteTopic

Il seguente esempio di codice mostra come utilizzare `DeleteTopic`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;
```

```
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.DeleteTopicRequest;
import software.amazon.awssdk.services.sns.model.DeleteTopicResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteTopic {
    public static void main(String[] args) {
        final String usage = ""

            Usage:    <topicArn>

            Where:
                topicArn - The ARN of the topic to delete.
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String topicArn = args[0];
        SnsClient snsClient = SnsClient.builder()
            .region(Region.US_EAST_1)
            .build();

        System.out.println("Deleting a topic with name: " + topicArn);
        deleteSNSTopic(snsClient, topicArn);
        snsClient.close();
    }

    public static void deleteSNSTopic(SnsClient snsClient, String topicArn) {
        try {
            DeleteTopicRequest request = DeleteTopicRequest.builder()
                .topicArn(topicArn)
                .build();
        }
    }
}
```

```

        DeleteTopicResponse result = snsClient.deleteTopic(request);
        System.out.println("\n\nStatus was " +
result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}

```

- Per i dettagli sull'API, consulta la [DeleteTopic](#) sezione AWS SDK for Java 2.x API Reference.

## GetSMSAttributes

Il seguente esempio di codice mostra come utilizzare `GetSMSAttributes`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.GetSubscriptionAttributesRequest;
import software.amazon.awssdk.services.sns.model.GetSubscriptionAttributesResponse;
import software.amazon.awssdk.services.sns.model.SnsException;
import java.util.Iterator;
import java.util.Map;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

```

```
public class GetSMSAttributes {
    public static void main(String[] args) {
        final String usage = ""

            Usage:    <topicArn>

            Where:
                topicArn - The ARN of the topic from which to retrieve
attributes.
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String topicArn = args[0];
        SnsClient snsClient = SnsClient.builder()
            .region(Region.US_EAST_1)
            .build();

        getSNSAttributes(snsClient, topicArn);
        snsClient.close();
    }

    public static void getSNSAttributes(SnsClient snsClient, String topicArn) {
        try {
            GetSubscriptionAttributesRequest request =
GetSubscriptionAttributesRequest.builder()
                .subscriptionArn(topicArn)
                .build();

            // Get the Subscription attributes
            GetSubscriptionAttributesResponse res =
snsClient.getSubscriptionAttributes(request);
            Map<String, String> map = res.attributes();

            // Iterate through the map
            Iterator iter = map.entrySet().iterator();
            while (iter.hasNext()) {
                Map.Entry entry = (Map.Entry) iter.next();
                System.out.println("[Key] : " + entry.getKey() + " [Value] : " +
entry.getValue());
            }
        }
    }
}
```



```
    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }

    System.out.println("\n\nStatus was good");
}
}
```

- Per informazioni dettagliate sulle API, consulta [GetSMSAttributes](#) nella Documentazione di riferimento per le API AWS SDK for Java 2.x .

## GetTopicAttributes

Il seguente esempio di codice mostra come usare `GetTopicAttributes`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.GetTopicAttributesRequest;
import software.amazon.awssdk.services.sns.model.GetTopicAttributesResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class GetTopicAttributes {
```

```
public static void main(String[] args) {
    final String usage = ""

        Usage:    <topicArn>

        Where:
            topicArn - The ARN of the topic to look up.
    """;

    if (args.length != 1) {
        System.out.println(usage);
        System.exit(1);
    }

    String topicArn = args[0];
    SnsClient snsClient = SnsClient.builder()
        .region(Region.US_EAST_1)
        .build();

    System.out.println("Getting attributes for a topic with name: " + topicArn);
    getSNSTopicAttributes(snsClient, topicArn);
    snsClient.close();
}

public static void getSNSTopicAttributes(SnsClient snsClient, String topicArn) {
    try {
        GetTopicAttributesRequest request = GetTopicAttributesRequest.builder()
            .topicArn(topicArn)
            .build();

        GetTopicAttributesResponse result =
snsClient.getTopicAttributes(request);
        System.out.println("\n\nStatus is " +
result.sdkHttpResponse().statusCode() + "\n\nAttributes: \n\n"
            + result.attributes());

    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Per i dettagli sull'API, consulta la [GetTopicAttributes](#) sezione AWS SDK for Java 2.x API Reference.

## ListPhoneNumbersOptedOut

Il seguente esempio di codice mostra come utilizzare `ListPhoneNumbersOptedOut`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.ListPhoneNumbersOptedOutRequest;
import software.amazon.awssdk.services.sns.model.ListPhoneNumbersOptedOutResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListOptOut {
    public static void main(String[] args) {
        SnsClient snsClient = SnsClient.builder()
            .region(Region.US_EAST_1)
            .build();

        listOpts(snsClient);
        snsClient.close();
    }

    public static void listOpts(SnsClient snsClient) {
        try {
```

```

        ListPhoneNumbersOptedOutRequest request =
ListPhoneNumbersOptedOutRequest.builder().build();
        ListPhoneNumbersOptedOutResponse result =
snsClient.listPhoneNumbersOptedOut(request);
        System.out.println("Status is " + result.sdkHttpResponse().statusCode()
+ "\n\nPhone Numbers: \n\n"
            + result.phoneNumbers());

    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}

```

- Per i dettagli sull'API, consulta la [ListPhoneNumbersOptedOut](#) sezione AWS SDK for Java 2.x API Reference.

## ListSubscriptions

Il seguente esempio di codice mostra come utilizzare `ListSubscriptions`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.ListSubscriptionsRequest;
import software.amazon.awssdk.services.sns.model.ListSubscriptionsResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 */

```

```
* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class ListSubscriptions {
    public static void main(String[] args) {
        SnsClient snsClient = SnsClient.builder()
            .region(Region.US_EAST_1)
            .build();

        listSNSSubscriptions(snsClient);
        snsClient.close();
    }

    public static void listSNSSubscriptions(SnsClient snsClient) {
        try {
            ListSubscriptionsRequest request = ListSubscriptionsRequest.builder()
                .build();

            ListSubscriptionsResponse result = snsClient.listSubscriptions(request);
            System.out.println(result.subscriptions());
        } catch (SnsException e) {

            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Per i dettagli sull'API, consulta la [ListSubscriptions](#) sezione AWS SDK for Java 2.x API Reference.

## ListTopics

Il seguente esempio di codice mostra come utilizzare `ListTopics`.

## SDK per Java 2.x

 Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.ListTopicsRequest;
import software.amazon.awssdk.services.sns.model.ListTopicsResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListTopics {
    public static void main(String[] args) {
        SnsClient snsClient = SnsClient.builder()
            .region(Region.US_EAST_1)
            .build();

        listSNSTopics(snsClient);
        snsClient.close();
    }

    public static void listSNSTopics(SnsClient snsClient) {
        try {
            ListTopicsRequest request = ListTopicsRequest.builder()
                .build();

            ListTopicsResponse result = snsClient.listTopics(request);
            System.out.println(
                "Status was " + result.sdkHttpResponse().statusCode() + "\n\n"
                + result.topics());
        } catch (SnsException e) {
            System.out.println("Error: " + e.getMessage());
        }
    }
}
```

```

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}

```

- Per i dettagli sull'API, consulta la [ListTopics](#) sezione AWS SDK for Java 2.x API Reference.

## Publish

Il seguente esempio di codice mostra come utilizzare `Publish`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.PublishRequest;
import software.amazon.awssdk.services.sns.model.PublishResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class PublishTopic {
    public static void main(String[] args) {
        final String usage = ""

                Usage:    <message> <topicArn>

```

```

        Where:
            message - The message text to send.
            topicArn - The ARN of the topic to publish.
        """;

    if (args.length != 2) {
        System.out.println(usage);
        System.exit(1);
    }

    String message = args[0];
    String topicArn = args[1];
    SnsClient snsClient = SnsClient.builder()
        .region(Region.US_EAST_1)
        .build();
    pubTopic(snsClient, message, topicArn);
    snsClient.close();
}

public static void pubTopic(SnsClient snsClient, String message, String
topicArn) {
    try {
        PublishRequest request = PublishRequest.builder()
            .message(message)
            .topicArn(topicArn)
            .build();

        PublishResponse result = snsClient.publish(request);
        System.out
            .println(result.messageId() + " Message sent. Status is " +
result.sdkHttpResponse().statusCode());

    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
}

```

- Per informazioni dettagliate sulle API, consulta [Pubblicazione](#) nella Documentazione di riferimento per le API AWS SDK for Java 2.x .



## SetSMSAttributes

Il seguente esempio di codice mostra come usare `SetSMSAttributes`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SetSmsAttributesRequest;
import software.amazon.awssdk.services.sns.model.SetSmsAttributesResponse;
import software.amazon.awssdk.services.sns.model.SnsException;
import java.util.HashMap;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class SetSMSAttributes {
    public static void main(String[] args) {
        HashMap<String, String> attributes = new HashMap<>(1);
        attributes.put("DefaultSMSType", "Transactional");
        attributes.put("UsageReportS3Bucket", "janbucket");

        SnsClient snsClient = SnsClient.builder()
            .region(Region.US_EAST_1)
            .build();
        setSNSAttributes(snsClient, attributes);
        snsClient.close();
    }

    public static void setSNSAttributes(SnsClient snsClient, HashMap<String, String>
attributes) {
        try {
```

```
        SetSmsAttributesRequest request = SetSmsAttributesRequest.builder()
            .attributes(attributes)
            .build();

        SetSmsAttributesResponse result = snsClient.setSMSAttributes(request);
        System.out.println("Set default Attributes to " + attributes + ". Status
was "
            + result.sdkHttpResponse().statusCode());

    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Per informazioni dettagliate sulle API, consulta [SetSMSAttributes](#) nella Documentazione di riferimento per le API AWS SDK for Java 2.x .

## SetSubscriptionAttributes

Il seguente esempio di codice mostra come usare `SetSubscriptionAttributes`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SnsException;
import java.util.ArrayList;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 */
```

```
* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class UseMessageFilterPolicy {
    public static void main(String[] args) {
        final String usage = ""

            Usage:    <subscriptionArn>

            Where:
                subscriptionArn - The ARN of a subscription.

            "";

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String subscriptionArn = args[0];
        SnsClient snsClient = SnsClient.builder()
            .region(Region.US_EAST_1)
            .build();

        usePolicy(snsClient, subscriptionArn);
        snsClient.close();
    }

    public static void usePolicy(SnsClient snsClient, String subscriptionArn) {
        try {
            SNSMessageFilterPolicy fp = new SNSMessageFilterPolicy();
            // Add a filter policy attribute with a single value
            fp.addAttribute("store", "example_corp");
            fp.addAttribute("event", "order_placed");

            // Add a prefix attribute
            fp.addAttributePrefix("customer_interests", "bas");

            // Add an anything-but attribute
            fp.addAttributeAnythingBut("customer_interests", "baseball");

            // Add a filter policy attribute with a list of values
            ArrayList<String> attributeValues = new ArrayList<>();
```

```
        attributeValues.add("rugby");
        attributeValues.add("soccer");
        attributeValues.add("hockey");
        fp.addAttribute("customer_interests", attributeValues);

        // Add a numeric attribute
        fp.addAttribute("price_usd", "=", 0);

        // Add a numeric attribute with a range
        fp.addAttributeRange("price_usd", ">", 0, "<=", 100);

        // Apply the filter policy attributes to an Amazon SNS subscription
        fp.apply(snsClient, subscriptionArn);

    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Per i dettagli sull'API, consulta la [SetSubscriptionAttributes](#) sezione AWS SDK for Java 2.x API Reference.

## SetTopicAttributes

Il seguente esempio di codice mostra come utilizzare `SetTopicAttributes`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SetTopicAttributesRequest;
import software.amazon.awssdk.services.sns.model.SetTopicAttributesResponse;
```

```
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class SetTopicAttributes {

    public static void main(String[] args) {
        final String usage = ""

            Usage:    <attribute> <topicArn> <value>

            Where:
                attribute - The attribute action to use. Valid parameters are:
Policy | DisplayName | DeliveryPolicy .
                topicArn - The ARN of the topic.\s
                value - The value for the attribute.
            """;

        if (args.length < 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String attribute = args[0];
        String topicArn = args[1];
        String value = args[2];

        SnsClient snsClient = SnsClient.builder()
            .region(Region.US_EAST_1)
            .build();

        setTopAttr(snsClient, attribute, topicArn, value);
        snsClient.close();
    }

    public static void setTopAttr(SnsClient snsClient, String attribute, String
topicArn, String value) {
        try {
```

```

        SetTopicAttributesRequest request = SetTopicAttributesRequest.builder()
            .attributeName(attribute)
            .attributeValue(value)
            .topicArn(topicArn)
            .build();

        SetTopicAttributesResponse result =
snsClient.setTopicAttributes(request);
        System.out.println(
            "\n\nStatus was " + result.sdkHttpResponse().statusCode() + "\n\nTopic " + request.topicArn()
                + " updated " + request.attributeName() + " to " +
            request.attributeValue());

    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
}

```

- Per i dettagli sull'API, consulta la [SetTopicAttributes](#) sezione AWS SDK for Java 2.x API Reference.

## Subscribe

Il seguente esempio di codice mostra come utilizzare `Subscribe`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Sottoscrivi un indirizzo email a un argomento.

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SnsException;

```

```
import software.amazon.awssdk.services.sns.model.SubscribeRequest;
import software.amazon.awssdk.services.sns.model.SubscribeResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class SubscribeEmail {
    public static void main(String[] args) {
        final String usage = ""
            Usage:    <topicArn> <email>

            Where:
                topicArn - The ARN of the topic to subscribe.
                email - The email address to use.
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String topicArn = args[0];
        String email = args[1];
        SnsClient snsClient = SnsClient.builder()
            .region(Region.US_EAST_1)
            .build();

        subEmail(snsClient, topicArn, email);
        snsClient.close();
    }

    public static void subEmail(SnsClient snsClient, String topicArn, String email)
    {
        try {
            SubscribeRequest request = SubscribeRequest.builder()
                .protocol("email")
                .endpoint(email)
                .returnSubscriptionArn(true)
                .topicArn(topicArn)
        }
    }
}
```

```

        .build();

        SubscribeResponse result = snsClient.subscribe(request);
        System.out.println("Subscription ARN: " + result.subscriptionArn() + "\n
\n Status is "
            + result.sdkHttpResponse().statusCode());

    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}

```

Sottoscrivi un endpoint HTTP a un argomento.

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.model.SubscribeRequest;
import software.amazon.awssdk.services.sns.model.SubscribeResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class SubscribeHTTPS {
    public static void main(String[] args) {
        final String usage = ""

            Usage:    <topicArn> <url>

            Where:
                topicArn - The ARN of the topic to subscribe.
                url - The HTTPS endpoint that you want to receive notifications.
            """;

        if (args.length < 2) {

```



```

        System.out.println(usage);
        System.exit(1);
    }

    String topicArn = args[0];
    String url = args[1];
    SnsClient snsClient = SnsClient.builder()
        .region(Region.US_EAST_1)
        .build();

    subHTTPS(snsClient, topicArn, url);
    snsClient.close();
}

public static void subHTTPS(SnsClient snsClient, String topicArn, String url) {
    try {
        SubscribeRequest request = SubscribeRequest.builder()
            .protocol("https")
            .endpoint(url)
            .returnSubscriptionArn(true)
            .topicArn(topicArn)
            .build();

        SubscribeResponse result = snsClient.subscribe(request);
        System.out.println("Subscription ARN is " + result.subscriptionArn() +
            "\n\n Status is "
                + result.sdkHttpResponse().statusCode());

    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}

```

Sottoscrivi una funzione Lambda a un argomento.

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.model.SubscribeRequest;
import software.amazon.awssdk.services.sns.model.SubscribeResponse;

```

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class SubscribeLambda {

    public static void main(String[] args) {

        final String usage = ""

            Usage:    <topicArn> <lambdaArn>

            Where:
                topicArn - The ARN of the topic to subscribe.
                lambdaArn - The ARN of an AWS Lambda function.
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String topicArn = args[0];
        String lambdaArn = args[1];
        SnsClient snsClient = SnsClient.builder()
            .region(Region.US_EAST_1)
            .build();

        String arnValue = subLambda(snsClient, topicArn, lambdaArn);
        System.out.println("Subscription ARN: " + arnValue);
        snsClient.close();
    }

    public static String subLambda(SnsClient snsClient, String topicArn, String
lambdaArn) {
        try {
            SubscribeRequest request = SubscribeRequest.builder()
                .protocol("lambda")
                .endpoint(lambdaArn)
```

```
        .returnSubscriptionArn(true)
        .topicArn(topicArn)
        .build();

    SubscribeResponse result = snsClient.subscribe(request);
    return result.subscriptionArn();

} catch (SnsException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
return "";
}
}
```

- Per informazioni dettagliate sulle API, consulta [Sottoscrizione](#) nella Documentazione di riferimento sulle API AWS SDK for Java 2.x .

## TagResource

Il seguente esempio di codice mostra come usare `TagResource`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.model.Tag;
import software.amazon.awssdk.services.sns.model.TagResourceRequest;
import java.util.ArrayList;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
```

```
* environment, including your credentials.
*
* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class AddTags {
    public static void main(String[] args) {
        final String usage = ""

            Usage:    <topicArn>

            Where:
                topicArn - The ARN of the topic to which tags are added.

            "";

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String topicArn = args[0];
        SnsClient snsClient = SnsClient.builder()
            .region(Region.US_EAST_1)
            .build();

        addTopicTags(snsClient, topicArn);
        snsClient.close();
    }

    public static void addTopicTags(SnsClient snsClient, String topicArn) {
        try {
            Tag tag = Tag.builder()
                .key("Team")
                .value("Development")
                .build();

            Tag tag2 = Tag.builder()
                .key("Environment")
                .value("Gamma")
                .build();

            List<Tag> tagList = new ArrayList<>();
```

```
        tagList.add(tag);
        tagList.add(tag2);

        TagResourceRequest tagResourceRequest = TagResourceRequest.builder()
            .resourceArn(topicArn)
            .tags(tagList)
            .build();

        snsClient.tagResource(tagResourceRequest);
        System.out.println("Tags have been added to " + topicArn);

    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Per i dettagli sull'API, consulta la [TagResource](#) sezione AWS SDK for Java 2.x API Reference.

## Unsubscribe

Il seguente esempio di codice mostra come utilizzare `Unsubscribe`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.model.UnsubscribeRequest;
import software.amazon.awssdk.services.sns.model.UnsubscribeResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 */
```

```
*
* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class Unsubscribe {
    public static void main(String[] args) {
        final String usage = ""

            Usage:    <subscriptionArn>

            Where:
                subscriptionArn - The ARN of the subscription to delete.
            "";

        if (args.length < 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String subscriptionArn = args[0];
        SnsClient snsClient = SnsClient.builder()
            .region(Region.US_EAST_1)
            .build();

        unSub(snsClient, subscriptionArn);
        snsClient.close();
    }

    public static void unSub(SnsClient snsClient, String subscriptionArn) {
        try {
            UnsubscribeRequest request = UnsubscribeRequest.builder()
                .subscriptionArn(subscriptionArn)
                .build();

            UnsubscribeResponse result = snsClient.unsubscribe(request);
            System.out.println("\n\nStatus was " +
result.sdkHttpResponse().statusCode()
                + "\n\nSubscription was removed for " +
request.subscriptionArn());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

```
    }  
  }  
}
```

- Per informazioni dettagliate sulle API, consulta [Annullamento della sottoscrizione](#) nella Documentazione di riferimento per le API AWS SDK for Java 2.x .

## Scenari

### Creazione di un endpoint di piattaforma per notifiche push

L'esempio di codice seguente mostra come creare un endpoint di piattaforma per notifiche push di Amazon SNS.

### SDK per Java 2.x

#### Note

C'è dell'altro GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;  
import software.amazon.awssdk.services.sns.SnsClient;  
import software.amazon.awssdk.services.sns.model.CreatePlatformEndpointRequest;  
import software.amazon.awssdk.services.sns.model.CreatePlatformEndpointResponse;  
import software.amazon.awssdk.services.sns.model.SnsException;  
  
/**  
 * Before running this Java V2 code example, set up your development  
 * environment, including your credentials.  
 *  
 * For more information, see the following documentation topic:  
 *  
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html  
 *  
 * In addition, create a platform application using the AWS Management Console.  
 * See this doc topic:  
 *  
 * https://docs.aws.amazon.com/sns/latest/dg/mobile-push-send-register.html  
 */
```

```
*
* Without the values created by following the previous link, this code examples
* does not work.
*/

public class RegistrationExample {
    public static void main(String[] args) {
        final String usage = ""

            Usage:      <token> <platformApplicationArn>

            Where:
                token - The name of the FIFO topic.\s
                platformApplicationArn - The ARN value of platform application.
You can get this value from the AWS Management Console.\s
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String token = args[0];
        String platformApplicationArn = args[1];
        SnsClient snsClient = SnsClient.builder()
            .region(Region.US_EAST_1)
            .build();

        createEndpoint(snsClient, token, platformApplicationArn);
    }

    public static void createEndpoint(SnsClient snsClient, String token, String
platformApplicationArn) {
        System.out.println("Creating platform endpoint with token " + token);
        try {
            CreatePlatformEndpointRequest endpointRequest =
CreatePlatformEndpointRequest.builder()
                .token(token)
                .platformApplicationArn(platformApplicationArn)
                .build();

            CreatePlatformEndpointResponse response =
snsClient.createPlatformEndpoint(endpointRequest);

```



```
        System.out.println("The ARN of the endpoint is " +
response.endpointArn());
    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

## Creazione e pubblicazione su un argomento FIFO

Gli esempi di codice seguenti mostrano come creare e pubblicare su un argomento FIFO Amazon SNS.

### SDK per Java 2.x

#### Note

C'è dell'altro GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

In questo esempio

- viene creato un argomento Amazon SNS FIFO, due code FIFO Amazon SQS e una coda Standard.
- viene effettuata la sottoscrizione all'argomento e pubblicato un messaggio nell'argomento.

Il [test](#) verifica la ricezione del messaggio in ogni coda. L'[esempio completo](#) mostra anche l'aggiunta di policy di accesso e l'eliminazione delle risorse alla fine.

```
public class PriceUpdateExample {
    public final static SnsClient snsClient = SnsClient.create();
    public final static SqsClient sqsClient = SqsClient.create();

    public static void main(String[] args) {

        final String usage = "\n" +
            "Usage: " +
```

```

        "    <topicName> <wholesaleQueueFifoName> <retailQueueFifoName>
<analyticsQueueName>\n\n" +
        "Where:\n" +
        "    fifoTopicName - The name of the FIFO topic that you want to
create. \n\n" +
        "    wholesaleQueueARN - The name of a SQS FIFO queue that will be
created for the wholesale consumer. \n\n"
        +
        "    retailQueueARN - The name of a SQS FIFO queue that will be created
for the retail consumer. \n\n" +
        "    analyticsQueueARN - The name of a SQS standard queue that will
be created for the analytics consumer. \n\n";
    if (args.length != 4) {
        System.out.println(usage);
        System.exit(1);
    }

    final String fifoTopicName = args[0];
    final String wholeSaleQueueName = args[1];
    final String retailQueueName = args[2];
    final String analyticsQueueName = args[3];

    // For convenience, the QueueData class holds metadata about a queue: ARN,
URL,
    // name and type.
    List<QueueData> queues = List.of(
        new QueueData(wholeSaleQueueName, QueueType.FIFO),
        new QueueData(retailQueueName, QueueType.FIFO),
        new QueueData(analyticsQueueName, QueueType.Standard));

    // Create queues.
    createQueues(queues);

    // Create a topic.
    String topicARN = createFIFOTopic(fifoTopicName);

    // Subscribe each queue to the topic.
    subscribeQueues(queues, topicARN);

    // Allow the newly created topic to send messages to the queues.
    addAccessPolicyToQueuesFINAL(queues, topicARN);

    // Publish a sample price update message with payload.

```

```
        publishPriceUpdate(topicARN, "{\"product\": 214, \"price\": 79.99}",
"Consumables");

        // Clean up resources.
        deleteSubscriptions(queues);
        deleteQueues(queues);
        deleteTopic(topicARN);
    }

    public static String createFIFOtopic(String topicName) {
        try {
            // Create a FIFO topic by using the SNS service client.
            Map<String, String> topicAttributes = Map.of(
                "FifoTopic", "true",
                "ContentBasedDeduplication", "false");

            CreateTopicRequest topicRequest = CreateTopicRequest.builder()
                .name(topicName)
                .attributes(topicAttributes)
                .build();

            CreateTopicResponse response = snsClient.createTopic(topicRequest);
            String topicArn = response.topicArn();
            System.out.println("The topic ARN is" + topicArn);

            return topicArn;

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }

    public static void subscribeQueues(List<QueueData> queues, String topicARN) {
        queues.forEach(queue -> {
            SubscribeRequest subscribeRequest = SubscribeRequest.builder()
                .topicArn(topicARN)
                .endpoint(queue.queueARN)
                .protocol("sqs")
                .build();

            // Subscribe to the endpoint by using the SNS service client.
```

```
        // Only Amazon SQS queues can receive notifications from an Amazon SNS
        FIFO
        // topic.
        SubscribeResponse subscribeResponse =
snsClient.subscribe(subscribeRequest);
        System.out.println("The queue [" + queue.queueARN + "] subscribed to the
topic [" + topicARN + "]);
        queue.subscriptionARN = subscribeResponse.subscriptionArn();
    });
}

    public static void publishPriceUpdate(String topicArn, String payload, String
groupId) {

        try {
            // Create and publish a message that updates the wholesale price.
            String subject = "Price Update";
            String dedupId = UUID.randomUUID().toString();
            String attributeName = "business";
            String attributeValue = "wholesale";

            MessageAttributeValue msgAttValue = MessageAttributeValue.builder()
                .dataType("String")
                .stringValue(attributeValue)
                .build();

            Map<String, MessageAttributeValue> attributes = new HashMap<>();
            attributes.put(attributeName, msgAttValue);
            PublishRequest pubRequest = PublishRequest.builder()
                .topicArn(topicArn)
                .subject(subject)
                .message(payload)
                .messageGroupId(groupId)
                .messageDeduplicationId(dedupId)
                .messageAttributes(attributes)
                .build();

            final PublishResponse response = snsClient.publish(pubRequest);
            System.out.println(response.messageId());
            System.out.println(response.sequenceNumber());
            System.out.println("Message was published to " + topicArn);

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }
    }
}
```

```
        System.exit(1);
    }
}
```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella Documentazione di riferimento delle API AWS SDK for Java 2.x .
  - [CreateTopic](#)
  - [Pubblicare](#)
  - [Subscribe](#)

## Pubblicazione di un messaggio in un argomento

Il codice di esempio seguente mostra come fare per:

- Creazione di un argomento Amazon SNS.
- Sottoscrivere un numero di telefono cellulare all'argomento.
- Pubblicazione di messaggi SMS nell'argomento in modo che tutti i numeri di telefono sottoscritti ricevano il messaggio in una sola volta.

## SDK per Java 2.x

### Note

C'è dell'altro GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Creare un argomento e restituire il suo ARN.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.CreateTopicRequest;
import software.amazon.awssdk.services.sns.model.CreateTopicResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 */
```

```
*
* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class CreateTopic {
    public static void main(String[] args) {
        final String usage = ""

            Usage:    <topicName>

            Where:
                topicName - The name of the topic to create (for example,
mytopic).

            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String topicName = args[0];
        System.out.println("Creating a topic with name: " + topicName);
        SnsClient snsClient = SnsClient.builder()
            .region(Region.US_EAST_1)
            .build();

        String arnVal = createSNSTopic(snsClient, topicName);
        System.out.println("The topic ARN is" + arnVal);
        snsClient.close();
    }

    public static String createSNSTopic(SnsClient snsClient, String topicName) {
        CreateTopicResponse result;
        try {
            CreateTopicRequest request = CreateTopicRequest.builder()
                .name(topicName)
                .build();

            result = snsClient.createTopic(request);
            return result.topicArn();
        } catch (SnsException e) {
```

```

        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
}

```

Sottoscrivere un endpoint a un argomento.

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.model.SubscribeRequest;
import software.amazon.awssdk.services.sns.model.SubscribeResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class SubscribeTextSMS {
    public static void main(String[] args) {
        final String usage = ""

                Usage:    <topicArn> <phoneNumber>

                Where:
                    topicArn - The ARN of the topic to subscribe.
                    phoneNumber - A mobile phone number that receives notifications
                    (for example, +1XXX5550100).
                """;

        if (args.length < 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String topicArn = args[0];
        String phoneNumber = args[1];
    }
}

```

```

        SnsClient snsClient = SnsClient.builder()
            .region(Region.US_EAST_1)
            .build();

        subTextSNS(snsClient, topicArn, phoneNumber);
        snsClient.close();
    }

    public static void subTextSNS(SnsClient snsClient, String topicArn, String
phoneNumber) {
        try {
            SubscribeRequest request = SubscribeRequest.builder()
                .protocol("sms")
                .endpoint(phoneNumber)
                .returnSubscriptionArn(true)
                .topicArn(topicArn)
                .build();

            SubscribeResponse result = snsClient.subscribe(request);
            System.out.println("Subscription ARN: " + result.subscriptionArn() + "\n
\n Status is "
                + result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}

```

Impostare gli attributi del messaggio, ad esempio l'ID del mittente, il prezzo massimo e il relativo tipo. Gli attributi del messaggio sono facoltativi.

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SetSmsAttributesRequest;
import software.amazon.awssdk.services.sns.model.SetSmsAttributesResponse;
import software.amazon.awssdk.services.sns.model.SnsException;
import java.util.HashMap;

/**
 * Before running this Java V2 code example, set up your development

```



```

* environment, including your credentials.
*
* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class SetSMSAttributes {
    public static void main(String[] args) {
        HashMap<String, String> attributes = new HashMap<>(1);
        attributes.put("DefaultSMSType", "Transactional");
        attributes.put("UsageReportS3Bucket", "janbucket");

        SnsClient snsClient = SnsClient.builder()
            .region(Region.US_EAST_1)
            .build();
        setSNSAttributes(snsClient, attributes);
        snsClient.close();
    }

    public static void setSNSAttributes(SnsClient snsClient, HashMap<String, String>
attributes) {
        try {
            SetSmsAttributesRequest request = SetSmsAttributesRequest.builder()
                .attributes(attributes)
                .build();

            SetSmsAttributesResponse result = snsClient.setSMSAttributes(request);
            System.out.println("Set default Attributes to " + attributes + ". Status
was "
                + result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}

```

Publicare un messaggio in un argomento. Il messaggio viene inviato a ogni abbonato.

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;

```

```
import software.amazon.awssdk.services.sns.model.PublishRequest;
import software.amazon.awssdk.services.sns.model.PublishResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class PublishTextSMS {
    public static void main(String[] args) {
        final String usage = ""

            Usage:    <message> <phoneNumber>

            Where:
                message - The message text to send.
                phoneNumber - The mobile phone number to which a message is sent
(for example, +1XXX5550100).\s
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String message = args[0];
        String phoneNumber = args[1];
        SnsClient snsClient = SnsClient.builder()
            .region(Region.US_EAST_1)
            .build();
        pubTextSMS(snsClient, message, phoneNumber);
        snsClient.close();
    }

    public static void pubTextSMS(SnsClient snsClient, String message, String
phoneNumber) {
        try {
            PublishRequest request = PublishRequest.builder()
                .message(message)
                .phoneNumber(phoneNumber)
```

```
        .build();

        PublishResponse result = snsClient.publish(request);
        System.out
            .println(result.messageId() + " Message sent. Status was " +
result.sdkHttpResponse().statusCode());

    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

## Pubblicazione di un SMS

Il seguente esempio di codice mostra come pubblicare messaggi SMS utilizzando Amazon SNS.

### SDK per Java 2.x

#### Note

C'è altro su [GitHub](#) Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.PublishRequest;
import software.amazon.awssdk.services.sns.model.PublishResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
```

```
public class PublishTextSMS {
    public static void main(String[] args) {
        final String usage = ""

            Usage:    <message> <phoneNumber>

            Where:
                message - The message text to send.
                phoneNumber - The mobile phone number to which a message is sent
(for example, +1XXX5550100).\s
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String message = args[0];
        String phoneNumber = args[1];
        SnsClient snsClient = SnsClient.builder()
            .region(Region.US_EAST_1)
            .build();
        pubTextSMS(snsClient, message, phoneNumber);
        snsClient.close();
    }

    public static void pubTextSMS(SnsClient snsClient, String message, String
phoneNumber) {
        try {
            PublishRequest request = PublishRequest.builder()
                .message(message)
                .phoneNumber(phoneNumber)
                .build();

            PublishResponse result = snsClient.publish(request);
            System.out
                .println(result.messageId() + " Message sent. Status was " +
result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

```
}
```

- Per informazioni dettagliate sulle API, consulta [Pubblicazione](#) nella Documentazione di riferimento per le API AWS SDK for Java 2.x .

## Esempi serverless

### Richiamo di una funzione Lambda da un trigger Amazon SNS

Il seguente esempio di codice mostra come implementare una funzione Lambda che riceve un evento attivato dalla ricezione di messaggi da un argomento SNS. La funzione recupera i messaggi dal parametro dell'evento e registra il contenuto di ogni messaggio.

### SDK per Java 2.x

#### Note

C'è altro su GitHub Trova l'esempio completo e scopri come eseguire la configurazione e l'esecuzione nel repository di [Esempi serverless](#).

### Consumo di un evento SNS con Lambda utilizzando Java.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package example;

import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.LambdaLogger;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.SNSEvent;
import com.amazonaws.services.lambda.runtime.events.SNSEvent.SNSRecord;

import java.util.Iterator;
import java.util.List;

public class SNSEventHandler implements RequestHandler<SNSEvent, Boolean> {
    LambdaLogger logger;
```

```
@Override
public Boolean handleRequest(SNSEvent event, Context context) {
    logger = context.getLogger();
    List<SNSRecord> records = event.getRecords();
    if (!records.isEmpty()) {
        Iterator<SNSRecord> recordsIter = records.iterator();
        while (recordsIter.hasNext()) {
            processRecord(recordsIter.next());
        }
    }
    return Boolean.TRUE;
}

public void processRecord(SNSRecord record) {
    try {
        String message = record.getSNS().getMessage();
        logger.log("message: " + message);
    } catch (Exception e) {
        throw new RuntimeException(e);
    }
}
}
```

## Esempi di Amazon SQS con SDK for Java 2.x

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando AWS SDK for Java 2.x con Amazon SQS.

Le operazioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le operazioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati e negli esempi tra servizi.

Scenari: esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio.

Ogni esempio include un collegamento a GitHub, dove puoi trovare istruzioni su come configurare ed eseguire il codice nel contesto.

## Nozioni di base

### Salve Amazon SQS

I seguenti esempi di codice mostrano come iniziare a usare Amazon SQS.

#### SDK per Java 2.x

##### Note

C'è altro su [GitHub](#) Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.SqsException;
import software.amazon.awssdk.services.sqs.paginators.ListQueuesIterable;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class HelloSQS {
    public static void main(String[] args) {
        SqsClient sqsClient = SqsClient.builder()
            .region(Region.US_WEST_2)
            .build();

        listQueues(sqsClient);
        sqsClient.close();
    }

    public static void listQueues(SqsClient sqsClient) {
        try {
            ListQueuesIterable listQueues = sqsClient.listQueuesPaginator();
            listQueues.stream()
                .flatMap(r -> r.queueUrls().stream())
        }
    }
}
```

```
        .forEach(content -> System.out.println(" Queue URL: " +
content.toLowerCase()));

    } catch (SqsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Per i dettagli sull'API, consulta la [ListQueues](#) sezione AWS SDK for Java 2.x API Reference.

## Argomenti

- [Azioni](#)
- [Scenari](#)
- [Esempi serverless](#)

## Azioni

### CreateQueue

Il seguente esempio di codice mostra come utilizzare `CreateQueue`.

## SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.ChangeMessageVisibilityRequest;
import software.amazon.awssdk.services.sqs.model.CreateQueueRequest;
import software.amazon.awssdk.services.sqs.model.DeleteMessageRequest;
import software.amazon.awssdk.services.sqs.model.GetQueueUrlRequest;
import software.amazon.awssdk.services.sqs.model.GetQueueUrlResponse;
import software.amazon.awssdk.services.sqs.model.ListQueuesRequest;
```



```
import software.amazon.awssdk.services.sqs.model.ListQueuesResponse;
import software.amazon.awssdk.services.sqs.model.Message;
import software.amazon.awssdk.services.sqs.model.ReceiveMessageRequest;
import software.amazon.awssdk.services.sqs.model.SendMessageBatchRequest;
import software.amazon.awssdk.services.sqs.model.SendMessageBatchRequestEntry;
import software.amazon.awssdk.services.sqs.model.SendMessageRequest;
import software.amazon.awssdk.services.sqs.model.SqsException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class SQSExample {
    public static void main(String[] args) {
        String queueName = "queue" + System.currentTimeMillis();
        SqsClient sqsClient = SqsClient.builder()
            .region(Region.US_WEST_2)
            .build();

        // Perform various tasks on the Amazon SQS queue.
        String queueUrl = createQueue(sqsClient, queueName);
        listQueues(sqsClient);
        listQueuesFilter(sqsClient, queueUrl);
        List<Message> messages = receiveMessages(sqsClient, queueUrl);
        sendBatchMessages(sqsClient, queueUrl);
        changeMessages(sqsClient, queueUrl, messages);
        deleteMessages(sqsClient, queueUrl, messages);
        sqsClient.close();
    }

    public static String createQueue(SqsClient sqsClient, String queueName) {
        try {
            System.out.println("\nCreate Queue");

            CreateQueueRequest createQueueRequest = CreateQueueRequest.builder()
                .queueName(queueName)
                .build();

            sqsClient.createQueue(createQueueRequest);
        }
    }
}
```

```
        System.out.println("\nGet queue url");

        GetQueueUrlResponse getQueueUrlResponse = sqsClient
.getQueueUrl(GetQueueUrlRequest.builder().queueName(queueName).build());
        return getQueueUrlResponse.queueUrl();

    } catch (SqsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}

public static void listQueues(SqsClient sqsClient) {

    System.out.println("\nList Queues");
    String prefix = "que";

    try {
        ListQueuesRequest listQueuesRequest =
ListQueuesRequest.builder().queueNamePrefix(prefix).build();
        ListQueuesResponse listQueuesResponse =
sqsClient.listQueues(listQueuesRequest);
        for (String url : listQueuesResponse.queueUrls()) {
            System.out.println(url);
        }

    } catch (SqsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void listQueuesFilter(SqsClient sqsClient, String queueUrl) {
    // List queues with filters
    String namePrefix = "queue";
    ListQueuesRequest filterListRequest = ListQueuesRequest.builder()
        .queueNamePrefix(namePrefix)
        .build();

    ListQueuesResponse listQueuesFilteredResponse =
sqsClient.listQueues(filterListRequest);
}
```

```
System.out.println("Queue URLs with prefix: " + namePrefix);
for (String url : listQueuesFilteredResponse.queueUrls()) {
    System.out.println(url);
}

System.out.println("\nSend message");
try {
    sqsClient.sendMessage(SendMessageRequest.builder()
        .queueUrl(queueUrl)
        .messageBody("Hello world!")
        .delaySeconds(10)
        .build());

} catch (SqsException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}

public static void sendBatchMessages(SqsClient sqsClient, String queueUrl) {

    System.out.println("\nSend multiple messages");
    try {
        SendMessageBatchRequest sendMessageBatchRequest =
SendMessageBatchRequest.builder()
            .queueUrl(queueUrl)

        .entries(SendMessageBatchRequestEntry.builder().id("id1").messageBody("Hello from
msg 1").build(),

SendMessageBatchRequestEntry.builder().id("id2").messageBody("msg
2").delaySeconds(10)

                .build())
            .build();
        sqsClient.sendMessageBatch(sendMessageBatchRequest);

    } catch (SqsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static List<Message> receiveMessages(SqsClient sqsClient, String
queueUrl) {
```

```
        System.out.println("\nReceive messages");
        try {
            ReceiveMessageRequest receiveMessageRequest =
ReceiveMessageRequest.builder()
                .queueUrl(queueUrl)
                .numberOfMessages(5)
                .build();
            return sqsClient.receiveMessage(receiveMessageRequest).messages();

        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return null;
    }

    public static void changeMessages(SqsClient sqsClient, String queueUrl,
List<Message> messages) {

        System.out.println("\nChange Message Visibility");
        try {

            for (Message message : messages) {
                ChangeMessageVisibilityRequest req =
ChangeMessageVisibilityRequest.builder()
                    .queueUrl(queueUrl)
                    .receiptHandle(message.receiptHandle())
                    .visibilityTimeout(100)
                    .build();
                sqsClient.changeMessageVisibility(req);
            }

        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void deleteMessages(SqsClient sqsClient, String queueUrl,
List<Message> messages) {
        System.out.println("\nDelete Messages");

        try {
```

```
        for (Message message : messages) {
            DeleteMessageRequest deleteMessageRequest =
DeleteMessageRequest.builder()
                .queueUrl(queueUrl)
                .receiptHandle(message.receiptHandle())
                .build();
            sqsClient.deleteMessage(deleteMessageRequest);
        }
    } catch (SqsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Per i dettagli sull'API, consulta la [CreateQueue](#) sezione AWS SDK for Java 2.x API Reference.

## DeleteMessage

Il seguente esempio di codice mostra come utilizzare `DeleteMessage`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
    try {
        for (Message message : messages) {
            DeleteMessageRequest deleteMessageRequest =
DeleteMessageRequest.builder()
                .queueUrl(queueUrl)
                .receiptHandle(message.receiptHandle())
                .build();
            sqsClient.deleteMessage(deleteMessageRequest);
        }
    } catch (SqsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }
```

```
        System.exit(1);
    }
```

- Per i dettagli sull'API, [DeleteMessage](#) consulta AWS SDK for Java 2.x API Reference.

## DeleteQueue

Il seguente esempio di codice mostra come utilizzare `DeleteQueue`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.GetQueueUrlRequest;
import software.amazon.awssdk.services.sqs.model.DeleteQueueRequest;
import software.amazon.awssdk.services.sqs.model.SqsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteQueue {
    public static void main(String[] args) {
        final String usage = ""

                Usage:    <queueName>

                Where:
                    queueName - The name of the Amazon SQS queue to delete.
    }
```

```
        """);

    if (args.length != 1) {
        System.out.println(usage);
        System.exit(1);
    }

    String queueName = args[0];
    SqsClient sqs = SqsClient.builder()
        .region(Region.US_WEST_2)
        .build();

    deleteSQSQueue(sqs, queueName);
    sqs.close();
}

public static void deleteSQSQueue(SqsClient sqsClient, String queueName) {
    try {
        GetQueueUrlRequest getQueueRequest = GetQueueUrlRequest.builder()
            .queueName(queueName)
            .build();

        String queueUrl = sqsClient.getQueueUrl(getQueueRequest).queueUrl();
        DeleteQueueRequest deleteQueueRequest = DeleteQueueRequest.builder()
            .queueUrl(queueUrl)
            .build();

        sqsClient.deleteQueue(deleteQueueRequest);

    } catch (SqsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Per i dettagli sull'API, [DeleteQueue](#) consulta AWS SDK for Java 2.x API Reference.

## GetQueueUrl

Il seguente esempio di codice mostra come utilizzare `GetQueueUrl`.

## SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
GetQueueUrlResponse getQueueUrlResponse = sqsClient
    .getQueueUrl(GetQueueUrlRequest.builder().queueName(queueName).build());
    return getQueueUrlResponse.queueUrl();
```

- Per i dettagli sull'API, [GetQueueUrl](#) consulta AWS SDK for Java 2.x API Reference.

## ListQueues

Il seguente esempio di codice mostra come utilizzare `ListQueues`.

## SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
String prefix = "que";

try {
    ListQueuesRequest listQueuesRequest =
ListQueuesRequest.builder().queueNamePrefix(prefix).build();
    ListQueuesResponse listQueuesResponse =
sqsClient.listQueues(listQueuesRequest);
    for (String url : listQueuesResponse.queueUrls()) {
        System.out.println(url);
    }
}
```



```
    } catch (SqsException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

- Per i dettagli sull'API, [ListQueues](#) consulta AWS SDK for Java 2.x API Reference.

## ReceiveMessage

Il seguente esempio di codice mostra come utilizzare `ReceiveMessage`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
    try {  
        ReceiveMessageRequest receiveMessageRequest =  
ReceiveMessageRequest.builder()  
            .queueUrl(queueUrl)  
            .numberOfMessages(5)  
            .build();  
        return sqsClient.receiveMessage(receiveMessageRequest).messages();  
    } catch (SqsException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
    return null;  
}
```

- Per i dettagli sull'API, [ReceiveMessage](#) consulta AWS SDK for Java 2.x API Reference.

## SendMessage

Il seguente esempio di codice mostra come utilizzare `SendMessage`.

## SDK per Java 2.x

 Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.CreateQueueRequest;
import software.amazon.awssdk.services.sqs.model.GetQueueUrlRequest;
import software.amazon.awssdk.services.sqs.model.SendMessageRequest;
import software.amazon.awssdk.services.sqs.model.SqsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class SendMessages {
    public static void main(String[] args) {
        final String usage = ""

                Usage:    <queueName> <message>

                Where:
                    queueName - The name of the queue.
                    message - The message to send.
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String queueName = args[0];
        String message = args[1];
```

```
SqsClient sqsClient = SqsClient.builder()
    .region(Region.US_WEST_2)
    .build();
sendMessage(sqsClient, queueName, message);
sqsClient.close();
}

public static void sendMessage(SqsClient sqsClient, String queueName, String
message) {
    try {
        CreateQueueRequest request = CreateQueueRequest.builder()
            .queueName(queueName)
            .build();
        sqsClient.createQueue(request);

        GetQueueUrlRequest getQueueRequest = GetQueueUrlRequest.builder()
            .queueName(queueName)
            .build();

        String queueUrl = sqsClient.getQueueUrl(getQueueRequest).queueUrl();
        SendMessageRequest sendMsgRequest = SendMessageRequest.builder()
            .queueUrl(queueUrl)
            .messageBody(message)
            .delaySeconds(5)
            .build();

        sqsClient.sendMessage(sendMsgRequest);


    } catch (SqsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Per i dettagli sull'API, [SendMessage](#) consulta AWS SDK for Java 2.x API Reference.

## SendMessageBatch

Il seguente esempio di codice mostra come utilizzare `SendMessageBatch`.

## SDK per Java 2.x

 Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
SendMessageBatchRequest sendMessageBatchRequest =
SendMessageBatchRequest.builder()
    .queueUrl(queueUrl)

    .entries(SendMessageBatchRequestEntry.builder().id("id1").messageBody("Hello from
msg 1").build(),

SendMessageBatchRequestEntry.builder().id("id2").messageBody("msg
2").delaySeconds(10)
        .build())
    .build();
sqsClient.sendMessageBatch(sendMessageBatchRequest);
```


- Per i dettagli sull'API, [SendMessageBatch](#) consulta AWS SDK for Java 2.x API Reference.

## Scenari

### Creazione e pubblicazione su un argomento FIFO

Gli esempi di codice seguenti mostrano come creare e pubblicare su un argomento FIFO Amazon SNS.

## SDK per Java 2.x

 Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

In questo esempio

- viene creato un argomento Amazon SNS FIFO, due code FIFO Amazon SQS e una coda Standard.
- viene effettuata la sottoscrizione all'argomento e pubblicato un messaggio nell'argomento.

Il [test](#) verifica la ricezione del messaggio in ogni coda. L'[esempio completo](#) mostra anche l'aggiunta di policy di accesso e l'eliminazione delle risorse alla fine.

```
public class PriceUpdateExample {
    public final static SnsClient snsClient = SnsClient.create();
    public final static SqsClient sqsClient = SqsClient.create();

    public static void main(String[] args) {

        final String usage = "\n" +
            "Usage: " +
            "    <topicName> <wholesaleQueueFifoName> <retailQueueFifoName>
<analyticsQueueName>\n\n" +
            "Where:\n" +
            "    fifoTopicName - The name of the FIFO topic that you want to
create. \n\n" +
            "    wholesaleQueueARN - The name of a SQS FIFO queue that will be
created for the wholesale consumer. \n\n"
            +
            "    retailQueueARN - The name of a SQS FIFO queue that will created
for the retail consumer. \n\n" +
            "    analyticsQueueARN - The name of a SQS standard queue that will
be created for the analytics consumer. \n\n";
        if (args.length != 4) {
            System.out.println(usage);
            System.exit(1);
        }

        final String fifoTopicName = args[0];
        final String wholeSaleQueueName = args[1];
        final String retailQueueName = args[2];
        final String analyticsQueueName = args[3];

        // For convenience, the QueueData class holds metadata about a queue: ARN,
URL,
        // name and type.
        List<QueueData> queues = List.of(
            new QueueData(wholeSaleQueueName, QueueType.FIFO),
            new QueueData(retailQueueName, QueueType.FIFO),
```

```
        new QueueData(analyticsQueueName, QueueType.Standard));

    // Create queues.
    createQueues(queues);

    // Create a topic.
    String topicARN = createFIFOTopic(fifoTopicName);

    // Subscribe each queue to the topic.
    subscribeQueues(queues, topicARN);

    // Allow the newly created topic to send messages to the queues.
    addAccessPolicyToQueuesFINAL(queues, topicARN);

    // Publish a sample price update message with payload.
    publishPriceUpdate(topicARN, "{\"product\": 214, \"price\": 79.99}",
"Consumables");

    // Clean up resources.
    deleteSubscriptions(queues);
    deleteQueues(queues);
    deleteTopic(topicARN);
}

public static String createFIFOTopic(String topicName) {
    try {
        // Create a FIFO topic by using the SNS service client.
        Map<String, String> topicAttributes = Map.of(
            "FifoTopic", "true",
            "ContentBasedDeduplication", "false");

        CreateTopicRequest topicRequest = CreateTopicRequest.builder()
            .name(topicName)
            .attributes(topicAttributes)
            .build();

        CreateTopicResponse response = snsClient.createTopic(topicRequest);
        String topicArn = response.topicArn();
        System.out.println("The topic ARN is" + topicArn);

        return topicArn;
    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }
}
```

```
        System.exit(1);
    }
    return "";
}

public static void subscribeQueues(List<QueueData> queues, String topicARN) {
    queues.forEach(queue -> {
        SubscribeRequest subscribeRequest = SubscribeRequest.builder()
            .topicArn(topicARN)
            .endpoint(queue.queueARN)
            .protocol("sqs")
            .build();

        // Subscribe to the endpoint by using the SNS service client.
        // Only Amazon SQS queues can receive notifications from an Amazon SNS
        FIFO
        // topic.
        SubscribeResponse subscribeResponse =
snsClient.subscribe(subscribeRequest);
        System.out.println("The queue [" + queue.queueARN + "] subscribed to the
topic [" + topicARN + "]);
        queue.subscriptionARN = subscribeResponse.subscriptionArn();
    });
}

public static void publishPriceUpdate(String topicArn, String payload, String
groupId) {

    try {
        // Create and publish a message that updates the wholesale price.
        String subject = "Price Update";
        String dedupId = UUID.randomUUID().toString();
        String attributeName = "business";
        String attributeValue = "wholesale";

        MessageAttributeValue msgAttValue = MessageAttributeValue.builder()
            .dataType("String")
            .stringValue(attributeValue)
            .build();

        Map<String, MessageAttributeValue> attributes = new HashMap<>();
        attributes.put(attributeName, msgAttValue);
        PublishRequest pubRequest = PublishRequest.builder()
            .topicArn(topicArn)
```

```
        .subject(subject)
        .message(payload)
        .messageGroupId(groupId)
        .messageDeduplicationId(dedupId)
        .messageAttributes(attributes)
        .build();

    final PublishResponse response = snsClient.publish(pubRequest);
    System.out.println(response.messageId());
    System.out.println(response.sequenceNumber());
    System.out.println("Message was published to " + topicArn);

} catch (SnsException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella Documentazione di riferimento delle API AWS SDK for Java 2.x .
  - [CreateTopic](#)
  - [Pubblicare](#)
  - [Subscribe](#)

## Esempi serverless

Richiamo di una funzione Lambda da un trigger Amazon SQS

Il seguente esempio di codice mostra come implementare una funzione Lambda che riceve un evento attivato dalla ricezione di messaggi da una coda SQS. La funzione recupera i messaggi dal parametro dell'evento e registra il contenuto di ogni messaggio.

SDK per Java 2.x

### Note

C'è altro su [GitHub](#) Trova l'esempio completo e scopri come eseguire la configurazione e l'esecuzione nel repository di [Esempi serverless](#).



## Utilizzo di un evento SQS con Lambda tramite Java.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.SQSEvent;
import com.amazonaws.services.lambda.runtime.events.SQSEvent.SQSMessage;

public class Function implements RequestHandler<SQSEvent, Void> {
    @Override
    public Void handleRequest(SQSEvent sqsEvent, Context context) {
        for (SQSMessage msg : sqsEvent.getRecords()) {
            processMessage(msg, context);
        }
        context.getLogger().log("done");
        return null;
    }

    private void processMessage(SQSMessage msg, Context context) {
        try {
            context.getLogger().log("Processed message " + msg.getBody());

            // TODO: Do interesting work based on the new message

        } catch (Exception e) {
            context.getLogger().log("An error occurred");
            throw e;
        }
    }
}
```

## Segnalazione di errori di elementi batch per funzioni Lambda con un trigger Amazon SQS

Il seguente esempio di codice mostra come implementare una risposta batch parziale per le funzioni Lambda che ricevono eventi da una coda SQS. La funzione riporta gli errori degli elementi batch nella risposta, segnalando a Lambda di riprovare tali messaggi in un secondo momento.

## SDK per Java 2.x

 Note

C'è di più su. GitHub Trova l'esempio completo e scopri come eseguire la configurazione e l'esecuzione nel repository di [Esempi serverless](#).

## Segnalazione di errori di elementi batch di SQS con Lambda tramite Java.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.SQSEvent;
import com.amazonaws.services.lambda.runtime.events.SQSBatchResponse;

import java.util.ArrayList;
import java.util.List;

public class ProcessSQSMessageBatch implements RequestHandler<SQSEvent,
SQSBatchResponse> {
    @Override
    public SQSBatchResponse handleRequest(SQSEvent sqsEvent, Context context) {

        List<SQSBatchResponse.BatchItemFailure> batchItemFailures = new
ArrayList<SQSBatchResponse.BatchItemFailure>();
        String messageId = "";
        for (SQSEvent.SQSMessage message : sqsEvent.getRecords()) {
            try {
                //process your message
                messageId = message.getMessageId();
            } catch (Exception e) {
                //Add failed message identifier to the batchItemFailures list
                batchItemFailures.add(new
SQSBatchResponse.BatchItemFailure(messageId));
            }
        }
        return new SQSBatchResponse(batchItemFailures);
    }
}
```

## Esempi di Step Functions con SDK for Java 2.x

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando AWS SDK for Java 2.x with Step Functions.

Le operazioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le operazioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati e negli esempi tra servizi.

Scenari: esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio.

Ogni esempio include un collegamento a GitHub, dove è possibile trovare istruzioni su come configurare ed eseguire il codice nel contesto.

Nozioni di base

Funzioni Hello Step

I seguenti esempi di codice mostrano come iniziare a usare Step Functions.

SDK per Java 2.x

### Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Versione Java di Hello.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sfn.SfnClient;
import software.amazon.awssdk.services.sfn.model.ListStateMachinesResponse;
import software.amazon.awssdk.services.sfn.model.SfnException;
import software.amazon.awssdk.services.sfn.model.StateMachineListItem;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
```

```
* environment, including your credentials.
*
* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class ListStateMachines {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        SfnClient sfnClient = SfnClient.builder()
            .region(region)
            .build();

        listMachines(sfnClient);
        sfnClient.close();
    }

    public static void listMachines(SfnClient sfnClient) {
        try {
            ListStateMachinesResponse response = sfnClient.listStateMachines();
            List<StateMachineListItem> machines = response.stateMachines();
            for (StateMachineListItem machine : machines) {
                System.out.println("The name of the state machine is: " +
machine.name());
                System.out.println("The ARN value is : " +
machine.stateMachineArn());
            }

        } catch (SfnException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Per i dettagli sull'API, [ListStateMachines](#) consulta AWS SDK for Java 2.x API Reference.

## Argomenti

- [Azioni](#)
- [Scenari](#)

## Azioni

### CreateActivity

Il seguente esempio di codice mostra come utilizzare `CreateActivity`.

SDK per Java 2.x

#### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public static String createActivity(SfnClient sfnClient, String activityName) {
    try {
        CreateActivityRequest activityRequest = CreateActivityRequest.builder()
            .name(activityName)
            .build();

        CreateActivityResponse response =
sfnClient.createActivity(activityRequest);
        return response.activityArn();

    } catch (SfnException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
```

- Per i dettagli sull'API, [CreateActivity](#) consulta AWS SDK for Java 2.x API Reference.

### CreateStateMachine

Il seguente esempio di codice mostra come utilizzare `CreateStateMachine`.

## SDK per Java 2.x

 Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public static String createMachine(SfnClient sfnClient, String roleARN, String
stateMachineName, String json) {
    try {
        CreateStateMachineRequest machineRequest =
CreateStateMachineRequest.builder()
            .definition(json)
            .name(stateMachineName)
            .roleArn(roleARN)
            .type(StateMachineType.STANDARD)
            .build();

        CreateStateMachineResponse response =
sfnClient.createStateMachine(machineRequest);
        return response.stateMachineArn();

    } catch (SfnException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
```

- Per i dettagli sull'API, [CreateStateMachine](#) consulta AWS SDK for Java 2.x API Reference.

## DeleteActivity

Il seguente esempio di codice mostra come utilizzare `DeleteActivity`.

## SDK per Java 2.x

**Note**

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public static void deleteActivity(SfnClient sfnClient, String actArn) {
    try {
        DeleteActivityRequest activityRequest = DeleteActivityRequest.builder()
            .activityArn(actArn)
            .build();

        sfnClient.deleteActivity(activityRequest);
        System.out.println("You have deleted " + actArn);

    } catch (SfnException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Per i dettagli sull'API, [DeleteActivity](#) consulta AWS SDK for Java 2.x API Reference.

**DeleteStateMachine**

Il seguente esempio di codice mostra come utilizzare `DeleteStateMachine`.

## SDK per Java 2.x

**Note**

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public static void deleteMachine(SfnClient sfnClient, String stateMachineArn) {
```

```
try {
    DeleteStateMachineRequest deleteStateMachineRequest =
DeleteStateMachineRequest.builder()
        .stateMachineArn(stateMachineArn)
        .build();

    sfnClient.deleteStateMachine(deleteStateMachineRequest);
    DescribeStateMachineRequest describeStateMachine =
DescribeStateMachineRequest.builder()
        .stateMachineArn(stateMachineArn)
        .build();

    while (true) {
        DescribeStateMachineResponse response =
sfnClient.describeStateMachine(describeStateMachine);
        System.out.println("The state machine is not deleted yet. The status
is " + response.status());
        Thread.sleep(3000);
    }

} catch (SfnException | InterruptedException e) {
    System.err.println(e.getMessage());
}
System.out.println(stateMachineArn + " was successfully deleted.");
}
```

- Per i dettagli sull'API, [DeleteStateMachine](#) consulta AWS SDK for Java 2.x API Reference.

## DescribeExecution

Il seguente esempio di codice mostra come utilizzare `DescribeExecution`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public static void describeExe(SfnClient sfnClient, String executionArn) {
```



```
try {
    DescribeExecutionRequest executionRequest =
DescribeExecutionRequest.builder()
        .executionArn(executionArn)
        .build();

    String status = "";
    boolean hasSucceeded = false;
    while (!hasSucceeded) {
        DescribeExecutionResponse response =
sfnClient.describeExecution(executionRequest);
        status = response.statusAsString();
        if (status.compareTo("RUNNING") == 0) {
            System.out.println("The state machine is still running, let's
wait for it to finish.");
            Thread.sleep(2000);
        } else if (status.compareTo("SUCCEEDED") == 0) {
            System.out.println("The Step Function workflow has succeeded");
            hasSucceeded = true;
        } else {
            System.out.println("The Status is neither running or
succeeded");
        }
    }
    System.out.println("The Status is " + status);

} catch (SfnException | InterruptedException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
}
```

- Per i dettagli sull'API, [DescribeExecution](#) consulta AWS SDK for Java 2.x API Reference.

## DescribeStateMachine

Il seguente esempio di codice mostra come utilizzare `DescribeStateMachine`.

## SDK per Java 2.x

 Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public static void describeStateMachine(SfnClient sfnClient, String
stateMachineArn) {
    try {
        DescribeStateMachineRequest stateMachineRequest =
DescribeStateMachineRequest.builder()
            .stateMachineArn(stateMachineArn)
            .build();

        DescribeStateMachineResponse response =
sfnClient.describeStateMachine(stateMachineRequest);
        System.out.println("The name of the State machine is " +
response.name());
        System.out.println("The status of the State machine is " +
response.status());
        System.out.println("The ARN value of the State machine is " +
response.stateMachineArn());
        System.out.println("The role ARN value is " + response.roleArn());

    } catch (SfnException e) {
        System.err.println(e.getMessage());
    }
}
```

- Per i dettagli sull'API, [DescribeStateMachine](#) consulta AWS SDK for Java 2.x API Reference.

## GetActivityTask

Il seguente esempio di codice mostra come utilizzare `GetActivityTask`.

## SDK per Java 2.x

 Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public static List<String> getActivityTask(SfnClient sfnClient, String actArn) {
    List<String> myList = new ArrayList<>();
    GetActivityTaskRequest getActivityTaskRequest =
    GetActivityTaskRequest.builder()
        .activityArn(actArn)
        .build();

    GetActivityTaskResponse response =
    sfnClient.getActivityTask(getActivityTaskRequest);
    myList.add(response.taskToken());
    myList.add(response.input());
    return myList;
}

/// <summary>
/// Stop execution of a Step Functions workflow.
/// </summary>
/// <param name="executionArn">The Amazon Resource Name (ARN) of
/// the Step Functions execution to stop.</param>
/// <returns>A Boolean value indicating the success of the action.</returns>
public async Task<bool> StopExecution(string executionArn)
{
    var response =
        await _amazonStepFunctions.StopExecutionAsync(new StopExecutionRequest
    { ExecutionArn = executionArn });
    return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
}
```

- Per i dettagli sull'API, [GetActivityTask](#) consulta AWS SDK for Java 2.x API Reference.

## ListActivities

Il seguente esempio di codice mostra come utilizzare `ListActivities`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sfn.SfnClient;
import software.amazon.awssdk.services.sfn.model.ListActivitiesRequest;
import software.amazon.awssdk.services.sfn.model.ListActivitiesResponse;
import software.amazon.awssdk.services.sfn.model.SfnException;
import software.amazon.awssdk.services.sfn.model.ActivityListItem;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListActivities {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        SfnClient sfnClient = SfnClient.builder()
            .region(region)
            .build();

        listAllActivites(sfnClient);
        sfnClient.close();
    }

    public static void listAllActivites(SfnClient sfnClient) {
        try {
            ListActivitiesRequest activitiesRequest =
                ListActivitiesRequest.builder()
```

```
        .maxResults(10)
        .build();

    ListActivitiesResponse response =
sfnClient.listActivities(activitiesRequest);
    List<ActivityListItem> items = response.activities();
    for (ActivityListItem item : items) {
        System.out.println("The activity ARN is " + item.activityArn());
        System.out.println("The activity name is " + item.name());
    }

    } catch (SfnException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Per i dettagli sull'API, [ListActivities](#) consulta AWS SDK for Java 2.x API Reference.

## ListExecutions

Il seguente esempio di codice mostra come utilizzare `ListExecutions`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public static void getExeHistory(SfnClient sfnClient, String exeARN) {
    try {
        GetExecutionHistoryRequest historyRequest =
GetExecutionHistoryRequest.builder()
        .executionArn(exeARN)
        .maxResults(10)
        .build();
```

```
        GetExecutionHistoryResponse historyResponse =
sfnClient.getExecutionHistory(historyRequest);
        List<HistoryEvent> events = historyResponse.events();
        for (HistoryEvent event : events) {
            System.out.println("The event type is " + event.type().toString());
        }

    } catch (SfnException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Per i dettagli sull'API, [ListExecutions](#) consulta AWS SDK for Java 2.x API Reference.

## ListStateMachines

Il seguente esempio di codice mostra come utilizzare `ListStateMachines`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sfn.SfnClient;
import software.amazon.awssdk.services.sfn.model.ListStateMachinesResponse;
import software.amazon.awssdk.services.sfn.model.SfnException;
import software.amazon.awssdk.services.sfn.model.StateMachineListItem;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 */
```

```
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class ListStateMachines {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        SfnClient sfnClient = SfnClient.builder()
            .region(region)
            .build();

        listMachines(sfnClient);
        sfnClient.close();
    }

    public static void listMachines(SfnClient sfnClient) {
        try {
            ListStateMachinesResponse response = sfnClient.listStateMachines();
            List<StateMachineListItem> machines = response.stateMachines();
            for (StateMachineListItem machine : machines) {
                System.out.println("The name of the state machine is: " +
machine.name());
                System.out.println("The ARN value is : " +
machine.stateMachineArn());
            }

        } catch (SfnException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Per i dettagli sull'API, [ListStateMachines](#) consulta AWS SDK for Java 2.x API Reference.

## SendTaskSuccess

Il seguente esempio di codice mostra come utilizzare `SendTaskSuccess`.

## SDK per Java 2.x

 Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public static void sendTaskSuccess(SfnClient sfnClient, String token, String
json) {
    try {
        SendTaskSuccessRequest successRequest = SendTaskSuccessRequest.builder()
            .taskToken(token)
            .output(json)
            .build();

        sfnClient.sendTaskSuccess(successRequest);


    } catch (SfnException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Per i dettagli sull'API, [SendTaskSuccess](#) consulta AWS SDK for Java 2.x API Reference.

## StartExecution

Il seguente esempio di codice mostra come utilizzare `StartExecution`.

## SDK per Java 2.x

 Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).



```
public static String startWorkflow(SfnClient sfnClient, String stateMachineArn,
String jsonEx) {
    UUID uuid = UUID.randomUUID();
    String uuidValue = uuid.toString();
    try {
        StartExecutionRequest executionRequest = StartExecutionRequest.builder()
            .input(jsonEx)
            .stateMachineArn(stateMachineArn)
            .name(uuidValue)
            .build();

        StartExecutionResponse response =
sfnClient.startExecution(executionRequest);
        return response.executionArn();

    } catch (SfnException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
```

- Per i dettagli sull'API, [StartExecution](#) consulta AWS SDK for Java 2.x API Reference.

## Scenari

Inizia a usare le macchine a stati

L'esempio di codice seguente mostra come:

- Crea un'attività.
- Crea una macchina a stati da una definizione di Amazon States Language che contiene l'attività creata in precedenza come passaggio.
- Esegui la macchina a stati e rispondi all'attività con l'input dell'utente.
- Ottieni lo stato e l'output finali al termine dell'esecuzione, quindi ripulisci le risorse.

## SDK per Java 2.x

 Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
/**
 * You can obtain the JSON file to create a state machine in the following
 * GitHub location.
 *
 * https://github.com/awsdocs/aws-doc-sdk-examples/tree/main/resources/sample_files
 *
 * To run this code example, place the chat_sfn_state_machine.json file into
 * your project's resources folder.
 *
 * Also, set up your development environment, including your credentials.
 *
 * For information, see this documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * This Java code example performs the following tasks:
 *
 * 1. Creates an activity.
 * 2. Creates a state machine.
 * 3. Describes the state machine.
 * 4. Starts execution of the state machine and interacts with it.
 * 5. Describes the execution.
 * 6. Delete the activity.
 * 7. Deletes the state machine.
 */
public class StepFunctionsScenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    public static void main(String[] args) throws Exception {
        final String usage = ""

                Usage:
                <roleARN> <activityName> <stateMachineName>
```

```

        Where:
            roleName - The name of the IAM role to create for this state
machine.
            activityName - The name of an activity to create.
            stateMachineName - The name of the state machine to create.
        """;

    if (args.length != 3) {
        System.out.println(usage);
        System.exit(1);
    }

    String roleName = args[0];
    String activityName = args[1];
    String stateMachineName = args[2];
    String polJSON = "{\n" +
        "    \"Version\": \"2012-10-17\",\n" +
        "    \"Statement\": [\n" +
        "        {\n" +
        "            \"Sid\": \"\",\n" +
        "            \"Effect\": \"Allow\",\n" +
        "            \"Principal\": {\n" +
        "                \"Service\": \"states.amazonaws.com\"\n" +
        "            },\n" +
        "            \"Action\": \"sts:AssumeRole\"\n" +
        "        }\n" +
        "    ]\n" +
        "}";

    Scanner sc = new Scanner(System.in);
    boolean action = false;

    Region region = Region.US_EAST_1;
    SfnClient sfnClient = SfnClient.builder()
        .region(region)
        .build();

    Region regionGl = Region.AWS_GLOBAL;
    IamClient iam = IamClient.builder()
        .region(regionGl)
        .build();

    System.out.println(DASHES);
    System.out.println("Welcome to the AWS Step Functions example scenario.");

```

```
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("1. Create an activity.");
String activityArn = createActivity(sfnClient, activityName);
System.out.println("The ARN of the activity is " + activityArn);
System.out.println(DASHES);

// Get JSON to use for the state machine and place the activityArn value
into
// it.
InputStream input = StepFunctionsScenario.class.getClassLoader()
    .getResourceAsStream("chat_sfn_state_machine.json");
ObjectMapper mapper = new ObjectMapper();
JsonNode jsonNode = mapper.readValue(input, JsonNode.class);
String jsonString = mapper.writeValueAsString(jsonNode);

// Modify the Resource node.
ObjectMapper objectMapper = new ObjectMapper();
JsonNode root = objectMapper.readTree(jsonString);
((ObjectNode) root.path("States").path("GetInput")).put("Resource",
activityArn);

// Convert the modified Java object back to a JSON string.
String stateDefinition = objectMapper.writeValueAsString(root);
System.out.println(stateDefinition);

System.out.println(DASHES);
System.out.println("2. Create a state machine.");
String roleARN = createIAMRole(iam, roleName, polJSON);
String stateMachineArn = createMachine(sfnClient, roleARN, stateMachineName,
stateDefinition);
System.out.println("The ARN of the state machine is " + stateMachineArn);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("3. Describe the state machine.");
describeStateMachine(sfnClient, stateMachineArn);
System.out.println("What should ChatSFN call you?");
String userName = sc.nextLine();
System.out.println("Hello " + userName);
System.out.println(DASHES);

System.out.println(DASHES);
```

```
// The JSON to pass to the StartExecution call.
String executionJson = "{ \"name\" : \"" + userName + "\" }";
System.out.println(executionJson);
System.out.println("4. Start execution of the state machine and interact
with it.");
String runArn = startWorkflow(sfnClient, stateMachineArn, executionJson);
System.out.println("The ARN of the state machine execution is " + runArn);
List<String> myList;
while (!action) {
    myList = getActivityTask(sfnClient, activityArn);
    System.out.println("ChatSFN: " + myList.get(1));
    System.out.println(userName + " please specify a value.");
    String myAction = sc.nextLine();
    if (myAction.compareTo("done") == 0)
        action = true;

    System.out.println("You have selected " + myAction);
    String taskJson = "{ \"action\" : \"" + myAction + "\" }";
    System.out.println(taskJson);
    sendTaskSuccess(sfnClient, myList.get(0), taskJson);
}
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("5. Describe the execution.");
describeExe(sfnClient, runArn);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("6. Delete the activity.");
deleteActivity(sfnClient, activityArn);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("7. Delete the state machines.");
deleteMachine(sfnClient, stateMachineArn);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("The AWS Step Functions example scenario is complete.");
System.out.println(DASHES);
}
```

```
public static String createIAMRole(IamClient iam, String rolename, String
polJSON) {
    try {
        CreateRoleRequest request = CreateRoleRequest.builder()
            .roleName(rolename)
            .assumeRolePolicyDocument(polJSON)
            .description("Created using the AWS SDK for Java")
            .build();

        CreateRoleResponse response = iam.createRole(request);
        return response.role().arn();

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}

public static void describeExe(SfnClient sfnClient, String executionArn) {
    try {
        DescribeExecutionRequest executionRequest =
DescribeExecutionRequest.builder()
            .executionArn(executionArn)
            .build();

        String status = "";
        boolean hasSucceeded = false;
        while (!hasSucceeded) {
            DescribeExecutionResponse response =
sfnClient.describeExecution(executionRequest);
            status = response.statusAsString();
            if (status.compareTo("RUNNING") == 0) {
                System.out.println("The state machine is still running, let's
wait for it to finish.");
                Thread.sleep(2000);
            } else if (status.compareTo("SUCCEEDED") == 0) {
                System.out.println("The Step Function workflow has succeeded");
                hasSucceeded = true;
            } else {
                System.out.println("The Status is neither running or
succeeded");
            }
        }
    }
}
```

```
        System.out.println("The Status is " + status);

    } catch (SfnException | InterruptedException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void sendTaskSuccess(SfnClient sfnClient, String token, String
json) {
    try {
        SendTaskSuccessRequest successRequest = SendTaskSuccessRequest.builder()
            .taskToken(token)
            .output(json)
            .build();

        sfnClient.sendTaskSuccess(successRequest);

    } catch (SfnException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static List<String> getActivityTask(SfnClient sfnClient, String actArn) {
    List<String> myList = new ArrayList<>();
    GetActivityTaskRequest getActivityTaskRequest =
GetActivityTaskRequest.builder()
        .activityArn(actArn)
        .build();

    GetActivityTaskResponse response =
sfnClient.getActivityTask(getActivityTaskRequest);
    myList.add(response.taskToken());
    myList.add(response.input());
    return myList;
}

public static void deleteActivity(SfnClient sfnClient, String actArn) {
    try {
        DeleteActivityRequest activityRequest = DeleteActivityRequest.builder()
            .activityArn(actArn)
            .build();
```

```
        sfnClient.deleteActivity(activityRequest);
        System.out.println("You have deleted " + actArn);

    } catch (SfnException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void describeStateMachine(SfnClient sfnClient, String
stateMachineArn) {
    try {
        DescribeStateMachineRequest stateMachineRequest =
DescribeStateMachineRequest.builder()
            .stateMachineArn(stateMachineArn)
            .build();

        DescribeStateMachineResponse response =
sfnClient.describeStateMachine(stateMachineRequest);
        System.out.println("The name of the State machine is " +
response.name());
        System.out.println("The status of the State machine is " +
response.status());
        System.out.println("The ARN value of the State machine is " +
response.stateMachineArn());
        System.out.println("The role ARN value is " + response.roleArn());

    } catch (SfnException e) {
        System.err.println(e.getMessage());
    }
}

public static void deleteMachine(SfnClient sfnClient, String stateMachineArn) {
    try {
        DeleteStateMachineRequest deleteStateMachineRequest =
DeleteStateMachineRequest.builder()
            .stateMachineArn(stateMachineArn)
            .build();

        sfnClient.deleteStateMachine(deleteStateMachineRequest);
        DescribeStateMachineRequest describeStateMachine =
DescribeStateMachineRequest.builder()
            .stateMachineArn(stateMachineArn)
            .build();
```



```
        while (true) {
            DescribeStateMachineResponse response =
sfnClient.describeStateMachine(describeStateMachine);
            System.out.println("The state machine is not deleted yet. The status
is " + response.status());
            Thread.sleep(3000);
        }

    } catch (SfnException | InterruptedException e) {
        System.err.println(e.getMessage());
    }
    System.out.println(stateMachineArn + " was successfully deleted.");
}

public static String startWorkflow(SfnClient sfnClient, String stateMachineArn,
String jsonEx) {
    UUID uuid = UUID.randomUUID();
    String uuidValue = uuid.toString();
    try {
        StartExecutionRequest executionRequest = StartExecutionRequest.builder()
            .input(jsonEx)
            .stateMachineArn(stateMachineArn)
            .name(uuidValue)
            .build();

        StartExecutionResponse response =
sfnClient.startExecution(executionRequest);
        return response.executionArn();

    } catch (SfnException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}

public static String createMachine(SfnClient sfnClient, String roleARN, String
stateMachineName, String json) {
    try {
        CreateStateMachineRequest machineRequest =
CreateStateMachineRequest.builder()
            .definition(json)
            .name(stateMachineName)
```

```
        .roleArn(roleARN)
        .type(StateMachineType.STANDARD)
        .build();

    CreateStateMachineResponse response =
sfnClient.createStateMachine(machineRequest);
    return response.stateMachineArn();

    } catch (SfnException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}

public static String createActivity(SfnClient sfnClient, String activityName) {
    try {
        CreateActivityRequest activityRequest = CreateActivityRequest.builder()
            .name(activityName)
            .build();

        CreateActivityResponse response =
sfnClient.createActivity(activityRequest);
        return response.activityArn();

    } catch (SfnException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
}
```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella Documentazione di riferimento delle API AWS SDK for Java 2.x .
  - [CreateActivity](#)
  - [CreateStateMachine](#)
  - [DeleteActivity](#)
  - [DeleteStateMachine](#)
  - [DescribeExecution](#)

- [DescribeStateMachine](#)
- [GetActivityTask](#)
- [ListActivities](#)
- [ListStateMachines](#)
- [SendTaskSuccess](#)
- [StartExecution](#)
- [StopExecution](#)

## AWS STS esempi che utilizzano SDK for Java 2.x

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando AWS SDK for Java 2.x with AWS STS.

Le operazioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le operazioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati e negli esempi tra servizi.

Scenari: esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio.

Ogni esempio include un collegamento a GitHub, dove è possibile trovare istruzioni su come configurare ed eseguire il codice nel contesto.

Argomenti

- [Azioni](#)

Azioni

### **AssumeRole**

Il seguente esempio di codice mostra come utilizzare `AssumeRole`.

## SDK per Java 2.x

 Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sts.StsClient;
import software.amazon.awssdk.services.sts.model.AssumeRoleRequest;
import software.amazon.awssdk.services.sts.model.StsException;
import software.amazon.awssdk.services.sts.model.AssumeRoleResponse;
import software.amazon.awssdk.services.sts.model.Credentials;
import java.time.Instant;
import java.time.ZoneId;
import java.time.format.DateTimeFormatter;
import java.time.format.FormatStyle;
import java.util.Locale;

/**
 * To make this code example work, create a Role that you want to assume.
 * Then define a Trust Relationship in the AWS Console. You can use this as an
 * example:
 *
 * {
 *   "Version": "2012-10-17",
 *   "Statement": [
 *     {
 *       "Effect": "Allow",
 *       "Principal": {
 *         "AWS": "<Specify the ARN of your IAM user you are using in this code
 * example>"
 *       },
 *       "Action": "sts:AssumeRole"
 *     }
 *   ]
 * }
 *
 * For more information, see "Editing the Trust Relationship for an Existing
 * Role" in the AWS Directory Service guide.
 *

```

```
* Also, set up your development environment, including your credentials.
*
* For information, see this documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class AssumeRole {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <roleArn> <roleSessionName>\s

            Where:
                roleArn - The Amazon Resource Name (ARN) of the role to assume
(for example, rn:aws:iam::000008047983:role/s3role).\s
                roleSessionName - An identifier for the assumed role session
(for example, mysession).\s
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String roleArn = args[0];
        String roleSessionName = args[1];
        Region region = Region.US_EAST_1;
        StsClient stsClient = StsClient.builder()
            .region(region)
            .build();

        assumeGivenRole(stsClient, roleArn, roleSessionName);
        stsClient.close();
    }

    public static void assumeGivenRole(StsClient stsClient, String roleArn, String
roleSessionName) {
        try {
            AssumeRoleRequest roleRequest = AssumeRoleRequest.builder()
                .roleArn(roleArn)
                .roleSessionName(roleSessionName)
                .build();

```

```
AssumeRoleResponse roleResponse = stsClient.assumeRole(roleRequest);
Credentials myCreds = roleResponse.credentials();

// Display the time when the temp creds expire.
Instant exTime = myCreds.expiration();
String tokenInfo = myCreds.sessionToken();

// Convert the Instant to readable date.
DateTimeFormatter formatter =
DateTimeFormatter.ofLocalizedDateTime(FormatStyle.SHORT)
    .withLocale(Locale.US)
    .withZone(ZoneId.systemDefault());

formatter.format(exTime);
System.out.println("The token " + tokenInfo + " expires on " + exTime);

} catch (StsException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
}
```

- Per i dettagli sull'API, [AssumeRole](#) consulta AWS SDK for Java 2.x API Reference.

## AWS Support esempi che utilizzano SDK for Java 2.x

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando AWS SDK for Java 2.x with AWS Support.

Le operazioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le operazioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati e negli esempi tra servizi.

Scenari: esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio.

Ogni esempio include un collegamento a GitHub, dove è possibile trovare istruzioni su come configurare ed eseguire il codice nel contesto.

Nozioni di base

## Salve AWS Support

L'esempio di codice seguente mostra come iniziare a utilizzare AWS Support.

### SDK per Java 2.x

#### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.support.SupportClient;
import software.amazon.awssdk.services.support.model.Category;
import software.amazon.awssdk.services.support.model.DescribeServicesRequest;
import software.amazon.awssdk.services.support.model.DescribeServicesResponse;
import software.amazon.awssdk.services.support.model.Service;
import software.amazon.awssdk.services.support.model.SupportException;
import java.util.ArrayList;
import java.util.List;

/**
 * Before running this Java (v2) code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * In addition, you must have the AWS Business Support Plan to use the AWS
 * Support Java API. For more information, see:
 *
 * https://aws.amazon.com/premiumsupport/plans/
 *
 * This Java example performs the following task:
 *
 * 1. Gets and displays available services.
 *
 * NOTE: To see multiple operations, see SupportScenario.
 */
```

```
public class HelloSupport {
    public static void main(String[] args) {
        Region region = Region.US_WEST_2;
        SupportClient supportClient = SupportClient.builder()
            .region(region)
            .build();

        System.out.println("***** Step 1. Get and display available services.");
        displayServices(supportClient);
    }

    // Return a List that contains a Service name and Category name.
    public static void displayServices(SupportClient supportClient) {
        try {
            DescribeServicesRequest servicesRequest =
                DescribeServicesRequest.builder()
                    .language("en")
                    .build();

            DescribeServicesResponse response =
                supportClient.describeServices(servicesRequest);
            List<Service> services = response.services();

            System.out.println("Get the first 10 services");
            int index = 1;
            for (Service service : services) {
                if (index == 11)
                    break;

                System.out.println("The Service name is: " + service.name());

                // Display the Categories for this service.
                List<Category> categories = service.categories();
                for (Category cat : categories) {
                    System.out.println("The category name is: " + cat.name());
                }
                index++;
            }
        } catch (SupportException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
}
```



```
}
```

- Per i dettagli sull'API, [DescribeServices](#) consulta AWS SDK for Java 2.x API Reference.

## Argomenti

- [Azioni](#)
- [Scenari](#)

## Azioni

### AddAttachmentsToSet

Il seguente esempio di codice mostra come utilizzare `AddAttachmentsToSet`.

SDK per Java 2.x

#### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public static String addAttachment(SupportClient supportClient, String
fileAttachment) {
    try {
        File myFile = new File(fileAttachment);
        InputStream sourceStream = new FileInputStream(myFile);
        SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);

        Attachment attachment = Attachment.builder()
            .fileName(myFile.getName())
            .data(sourceBytes)
            .build();

        AddAttachmentsToSetRequest setRequest =
AddAttachmentsToSetRequest.builder()
            .attachments(attachment)
            .build();
```

```
        AddAttachmentsToSetResponse response =
supportClient.addAttachmentsToSet(setRequest);
        return response.attachmentSetId();

    } catch (SupportException | FileNotFoundException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
    return "";
}
```

- Per i dettagli sull'API, [AddAttachmentsToSet](#) consulta AWS SDK for Java 2.x API Reference.

## AddCommunicationToCase

Il seguente esempio di codice mostra come utilizzare `AddCommunicationToCase`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public static void addAttachSupportCase(SupportClient supportClient, String
caseId, String attachmentSetId) {
    try {
        AddCommunicationToCaseRequest caseRequest =
AddCommunicationToCaseRequest.builder()
            .caseId(caseId)
            .attachmentSetId(attachmentSetId)
            .communicationBody("Please refer to attachment for details.")
            .build();

        AddCommunicationToCaseResponse response =
supportClient.addCommunicationToCase(caseRequest);
        if (response.result())
            System.out.println("You have successfully added a communication to
an AWS Support case");
        else
```

```
        System.out.println("There was an error adding the communication to
an AWS Support case");

    } catch (SupportException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}
```

- Per i dettagli sull'API, [AddCommunicationToCase](#) consulta AWS SDK for Java 2.x API Reference.

## CreateCase

Il seguente esempio di codice mostra come utilizzare `CreateCase`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public static String createSupportCase(SupportClient supportClient, List<String>
sevCatList, String sevLevel) {
    try {
        String serviceCode = sevCatList.get(0);
        String caseCat = sevCatList.get(1);
        CreateCaseRequest caseRequest = CreateCaseRequest.builder()
            .categoryCode(caseCat.toLowerCase())
            .serviceCode(serviceCode.toLowerCase())
            .severityCode(sevLevel.toLowerCase())
            .communicationBody("Test issue with " +
serviceCode.toLowerCase())
            .subject("Test case, please ignore")
            .language("en")
            .issueType("technical")
            .build();
```

```
        CreateCaseResponse response = supportClient.createCase(caseRequest);
        return response.caseId();

    } catch (SupportException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
    return "";
}
```

- Per i dettagli sull'API, [CreateCase](#) consulta AWS SDK for Java 2.x API Reference.

## DescribeAttachment

Il seguente esempio di codice mostra come utilizzare `DescribeAttachment`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public static void describeAttachment(SupportClient supportClient, String
attachId) {
    try {
        DescribeAttachmentRequest attachmentRequest =
DescribeAttachmentRequest.builder()
            .attachmentId(attachId)
            .build();

        DescribeAttachmentResponse response =
supportClient.describeAttachment(attachmentRequest);
        System.out.println("The name of the file is " +
response.attachment().fileName());

    } catch (SupportException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}
```

```
}
```

- Per i dettagli sull'API, [DescribeAttachment](#) consulta AWS SDK for Java 2.x API Reference.

## DescribeCases

Il seguente esempio di codice mostra come utilizzare `DescribeCases`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public static void getOpenCase(SupportClient supportClient) {
    try {
        // Specify the start and end time.
        Instant now = Instant.now();
        java.time.LocalDate.now();
        Instant yesterday = now.minus(1, ChronoUnit.DAYS);

        DescribeCasesRequest describeCasesRequest =
DescribeCasesRequest.builder()
            .maxResults(20)
            .afterTime(yesterday.toString())
            .beforeTime(now.toString())
            .build();

        DescribeCasesResponse response =
supportClient.describeCases(describeCasesRequest);
        List<CaseDetails> cases = response.cases();
        for (CaseDetails sinCase : cases) {
            System.out.println("The case status is " + sinCase.status());
            System.out.println("The case Id is " + sinCase.caseId());
            System.out.println("The case subject is " + sinCase.subject());
        }

    } catch (SupportException e) {
        System.out.println(e.getLocalizedMessage());
    }
}
```

```
        System.exit(1);
    }
}
```

- Per i dettagli sull'API, [DescribeCases](#) consulta AWS SDK for Java 2.x API Reference.

## DescribeCommunications

Il seguente esempio di codice mostra come utilizzare `DescribeCommunications`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public static String listCommunications(SupportClient supportClient, String
caseId) {
    try {
        String attachId = null;
        DescribeCommunicationsRequest communicationsRequest =
DescribeCommunicationsRequest.builder()
            .caseId(caseId)
            .maxResults(10)
            .build();

        DescribeCommunicationsResponse response =
supportClient.describeCommunications(communicationsRequest);
        List<Communication> communications = response.communications();
        for (Communication comm : communications) {
            System.out.println("the body is: " + comm.body());

            // Get the attachment id value.
            List<AttachmentDetails> attachments = comm.attachmentSet();
            for (AttachmentDetails detail : attachments) {
                attachId = detail.attachmentId();
            }
        }
        return attachId;
    }
}
```

```
    } catch (SupportException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
    return "";
}
```

- Per i dettagli sull'API, [DescribeCommunications](#) consulta AWS SDK for Java 2.x API Reference.

## DescribeServices

Il seguente esempio di codice mostra come utilizzare `DescribeServices`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
// Return a List that contains a Service name and Category name.
public static List<String> displayServices(SupportClient supportClient) {
    try {
        DescribeServicesRequest servicesRequest =
DescribeServicesRequest.builder()
            .language("en")
            .build();

        DescribeServicesResponse response =
supportClient.describeServices(servicesRequest);
        String serviceCode = null;
        String catName = null;
        List<String> sevCatList = new ArrayList<>();
        List<Service> services = response.services();

        System.out.println("Get the first 10 services");
        int index = 1;
        for (Service service : services) {
            if (index == 11)
```

```
        break;

        System.out.println("The Service name is: " + service.name());
        if (service.name().compareTo("Account") == 0)
            serviceCode = service.code();

        // Get the Categories for this service.
        List<Category> categories = service.categories();
        for (Category cat : categories) {
            System.out.println("The category name is: " + cat.name());
            if (cat.name().compareTo("Security") == 0)
                catName = cat.name();
        }
        index++;
    }

    // Push the two values to the list.
    sevCatList.add(serviceCode);
    sevCatList.add(catName);
    return sevCatList;

} catch (SupportException e) {
    System.out.println(e.getLocalizedMessage());
    System.exit(1);
}
return null;
}
```

- Per i dettagli sull'API, [DescribeServices](#) consulta AWS SDK for Java 2.x API Reference.

## DescribeSeverityLevels

Il seguente esempio di codice mostra come utilizzare `DescribeSeverityLevels`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).



```
public static String displaySevLevels(SupportClient supportClient) {
    try {
        DescribeSeverityLevelsRequest severityLevelsRequest =
DescribeSeverityLevelsRequest.builder()
            .language("en")
            .build();

        DescribeSeverityLevelsResponse response =
supportClient.describeSeverityLevels(severityLevelsRequest);
        List<SeverityLevel> severityLevels = response.severityLevels();
        String levelName = null;
        for (SeverityLevel sevLevel : severityLevels) {
            System.out.println("The severity level name is: " +
sevLevel.name());
            if (sevLevel.name().compareTo("High") == 0)
                levelName = sevLevel.name();
        }
        return levelName;

    } catch (SupportException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
    return "";
}
```

- Per i dettagli sull'API, [DescribeSeverityLevels](#) consulta AWS SDK for Java 2.x API Reference.

## ResolveCase

Il seguente esempio di codice mostra come utilizzare `ResolveCase`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public static void resolveSupportCase(SupportClient supportClient, String
caseId) {
    try {
        ResolveCaseRequest caseRequest = ResolveCaseRequest.builder()
            .caseId(caseId)
            .build();

        ResolveCaseResponse response = supportClient.resolveCase(caseRequest);
        System.out.println("The status of case " + caseId + " is " +
response.finalCaseStatus());

    } catch (SupportException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}
```

- Per i dettagli sull'API, [ResolveCase](#) consulta AWS SDK for Java 2.x API Reference.

## Scenari

### Come iniziare con i casi

L'esempio di codice seguente mostra come:

- Ottieni e visualizza i servizi e i livelli di gravità disponibili per i casi.
- Crea una richiesta di supporto utilizzando un servizio, una categoria e un livello di gravità selezionato.
- Ottieni e visualizza un elenco di casi aperti per il giorno corrente.
- Aggiungi un set di collegamenti e una comunicazione al nuovo caso.
- Descrivi il nuovo collegamento e la nuova comunicazione per il caso.
- Risolvi il caso.
- Ottieni e visualizza un elenco di casi risolti per il giorno corrente.

## SDK per Java 2.x

 Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Esegui varie AWS Support operazioni.

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.support.SupportClient;
import software.amazon.awssdk.services.support.model.AddAttachmentsToSetResponse;
import software.amazon.awssdk.services.support.model.AddCommunicationToCaseRequest;
import software.amazon.awssdk.services.support.model.AddCommunicationToCaseResponse;
import software.amazon.awssdk.services.support.model.Attachment;
import software.amazon.awssdk.services.support.model.AttachmentDetails;
import software.amazon.awssdk.services.support.model.CaseDetails;
import software.amazon.awssdk.services.support.model.Category;
import software.amazon.awssdk.services.support.model.Communication;
import software.amazon.awssdk.services.support.model.CreateCaseRequest;
import software.amazon.awssdk.services.support.model.CreateCaseResponse;
import software.amazon.awssdk.services.support.model.DescribeAttachmentRequest;
import software.amazon.awssdk.services.support.model.DescribeAttachmentResponse;
import software.amazon.awssdk.services.support.model.DescribeCasesRequest;
import software.amazon.awssdk.services.support.model.DescribeCasesResponse;
import software.amazon.awssdk.services.support.model.DescribeCommunicationsRequest;
import software.amazon.awssdk.services.support.model.DescribeCommunicationsResponse;
import software.amazon.awssdk.services.support.model.DescribeServicesRequest;
import software.amazon.awssdk.services.support.model.DescribeServicesResponse;
import software.amazon.awssdk.services.support.model.DescribeSeverityLevelsRequest;
import software.amazon.awssdk.services.support.model.DescribeSeverityLevelsResponse;
import software.amazon.awssdk.services.support.model.ResolveCaseRequest;
import software.amazon.awssdk.services.support.model.ResolveCaseResponse;
import software.amazon.awssdk.services.support.model.Service;
import software.amazon.awssdk.services.support.model.SeverityLevel;
import software.amazon.awssdk.services.support.model.SupportException;
import software.amazon.awssdk.services.support.model.AddAttachmentsToSetRequest;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
```

```
import java.time.Instant;
import java.time.temporal.ChronoUnit;
import java.util.ArrayList;
import java.util.List;

/**
 * Before running this Java (v2) code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * In addition, you must have the AWS Business Support Plan to use the AWS
 * Support Java API. For more information, see:
 *
 * https://aws.amazon.com/premiumsupport/plans/
 *
 * This Java example performs the following tasks:
 *
 * 1. Gets and displays available services.
 * 2. Gets and displays severity levels.
 * 3. Creates a support case by using the selected service, category, and
 * severity level.
 * 4. Gets a list of open cases for the current day.
 * 5. Creates an attachment set with a generated file.
 * 6. Adds a communication with the attachment to the support case.
 * 7. Lists the communications of the support case.
 * 8. Describes the attachment set included with the communication.
 * 9. Resolves the support case.
 * 10. Gets a list of resolved cases for the current day.
 */
public class SupportScenario {

    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    public static void main(String[] args) {
        final String usage = ""

            Usage:
            <fileAttachment>Where:
            fileAttachment - The file can be a simple saved .txt file to use
            as an email attachment.\s
            """;
    }
}
```

```
    if (args.length != 1) {
        System.out.println(usage);
        System.exit(1);
    }

    String fileAttachment = args[0];
    Region region = Region.US_WEST_2;
    SupportClient supportClient = SupportClient.builder()
        .region(region)
        .build();

    System.out.println(DASHES);
    System.out.println("***** Welcome to the AWS Support case example
scenario.");
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("1. Get and display available services.");
    List<String> sevCatList = displayServices(supportClient);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("2. Get and display Support severity levels.");
    String sevLevel = displaySevLevels(supportClient);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("3. Create a support case using the selected service,
category, and severity level.");
    String caseId = createSupportCase(supportClient, sevCatList, sevLevel);
    if (caseId.compareTo("") == 0) {
        System.out.println("A support case was not successfully created!");
        System.exit(1);
    } else
        System.out.println("Support case " + caseId + " was successfully
created!");
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("4. Get open support cases.");
    getOpenCase(supportClient);
    System.out.println(DASHES);
```

```
        System.out.println(DASHES);
        System.out.println("5. Create an attachment set with a generated file to add
to the case.");
        String attachmentSetId = addAttachment(supportClient, fileAttachment);
        System.out.println("The Attachment Set id value is" + attachmentSetId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6. Add communication with the attachment to the support
case.");
        addAttachSupportCase(supportClient, caseId, attachmentSetId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. List the communications of the support case.");
        String attachId = listCommunications(supportClient, caseId);
        System.out.println("The Attachment id value is" + attachId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("8. Describe the attachment set included with the
communication.");
        describeAttachment(supportClient, attachId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("9. Resolve the support case.");
        resolveSupportCase(supportClient, caseId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("10. Get a list of resolved cases for the current day.");
        getResolvedCase(supportClient);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("***** This Scenario has successfully completed");
        System.out.println(DASHES);
    }

    public static void getResolvedCase(SupportClient supportClient) {
        try {
            // Specify the start and end time.
            Instant now = Instant.now();
```

```
        java.time.LocalDate.now();
        Instant yesterday = now.minus(1, ChronoUnit.DAYS);

        DescribeCasesRequest describeCasesRequest =
DescribeCasesRequest.builder()
        .maxResults(30)
        .afterTime(yesterday.toString())
        .beforeTime(now.toString())
        .includeResolvedCases(true)
        .build();

        DescribeCasesResponse response =
supportClient.describeCases(describeCasesRequest);
        List<CaseDetails> cases = response.cases();
        for (CaseDetails sinCase : cases) {
            if (sinCase.status().compareTo("resolved") == 0)
                System.out.println("The case status is " + sinCase.status());
        }

    } catch (SupportException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}

public static void resolveSupportCase(SupportClient supportClient, String
caseId) {
    try {
        ResolveCaseRequest caseRequest = ResolveCaseRequest.builder()
            .caseId(caseId)
            .build();

        ResolveCaseResponse response = supportClient.resolveCase(caseRequest);
        System.out.println("The status of case " + caseId + " is " +
response.finalCaseStatus());

    } catch (SupportException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}

public static void describeAttachment(SupportClient supportClient, String
attachId) {
```

```
        try {
            DescribeAttachmentRequest attachmentRequest =
DescribeAttachmentRequest.builder()
                .attachmentId(attachId)
                .build();

            DescribeAttachmentResponse response =
supportClient.describeAttachment(attachmentRequest);
            System.out.println("The name of the file is " +
response.attachment().fileName());

        } catch (SupportException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    public static String listCommunications(SupportClient supportClient, String
caseId) {
        try {
            String attachId = null;
            DescribeCommunicationsRequest communicationsRequest =
DescribeCommunicationsRequest.builder()
                .caseId(caseId)
                .maxResults(10)
                .build();

            DescribeCommunicationsResponse response =
supportClient.describeCommunications(communicationsRequest);
            List<Communication> communications = response.communications();
            for (Communication comm : communications) {
                System.out.println("the body is: " + comm.body());

                // Get the attachment id value.
                List<AttachmentDetails> attachments = comm.attachmentSet();
                for (AttachmentDetails detail : attachments) {
                    attachId = detail.attachmentId();
                }
            }
            return attachId;

        } catch (SupportException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
}
```



```
    }
    return "";
}

public static void addAttachSupportCase(SupportClient supportClient, String
caseId, String attachmentSetId) {
    try {
        AddCommunicationToCaseRequest caseRequest =
AddCommunicationToCaseRequest.builder()
            .caseId(caseId)
            .attachmentSetId(attachmentSetId)
            .communicationBody("Please refer to attachment for details.")
            .build();

        AddCommunicationToCaseResponse response =
supportClient.addCommunicationToCase(caseRequest);
        if (response.result())
            System.out.println("You have successfully added a communication to
an AWS Support case");
        else
            System.out.println("There was an error adding the communication to
an AWS Support case");

    } catch (SupportException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}

public static String addAttachment(SupportClient supportClient, String
fileAttachment) {
    try {
        File myFile = new File(fileAttachment);
        InputStream sourceStream = new FileInputStream(myFile);
        SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);

        Attachment attachment = Attachment.builder()
            .fileName(myFile.getName())
            .data(sourceBytes)
            .build();

        AddAttachmentsToSetRequest setRequest =
AddAttachmentsToSetRequest.builder()
            .attachments(attachment)
```

```
        .build());

        AddAttachmentsToSetResponse response =
supportClient.addAttachmentsToSet(setRequest);
        return response.attachmentSetId();

    } catch (SupportException | FileNotFoundException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
    return "";
}

public static void getOpenCase(SupportClient supportClient) {
    try {
        // Specify the start and end time.
        Instant now = Instant.now();
        java.time.LocalDate.now();
        Instant yesterday = now.minus(1, ChronoUnit.DAYS);

        DescribeCasesRequest describeCasesRequest =
DescribeCasesRequest.builder()
            .maxResults(20)
            .afterTime(yesterday.toString())
            .beforeTime(now.toString())
            .build();

        DescribeCasesResponse response =
supportClient.describeCases(describeCasesRequest);
        List<CaseDetails> cases = response.cases();
        for (CaseDetails sinCase : cases) {
            System.out.println("The case status is " + sinCase.status());
            System.out.println("The case Id is " + sinCase.caseId());
            System.out.println("The case subject is " + sinCase.subject());
        }

    } catch (SupportException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}

public static String createSupportCase(SupportClient supportClient, List<String>
sevCatList, String sevLevel) {
```

```
    try {
        String serviceCode = sevCatList.get(0);
        String caseCat = sevCatList.get(1);
        CreateCaseRequest caseRequest = CreateCaseRequest.builder()
            .categoryCode(caseCat.toLowerCase())
            .serviceCode(serviceCode.toLowerCase())
            .severityCode(sevLevel.toLowerCase())
            .communicationBody("Test issue with " +
serviceCode.toLowerCase())
            .subject("Test case, please ignore")
            .language("en")
            .issueType("technical")
            .build();

        CreateCaseResponse response = supportClient.createCase(caseRequest);
        return response.caseId();

    } catch (SupportException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
    return "";
}

public static String displaySevLevels(SupportClient supportClient) {
    try {
        DescribeSeverityLevelsRequest severityLevelsRequest =
DescribeSeverityLevelsRequest.builder()
            .language("en")
            .build();

        DescribeSeverityLevelsResponse response =
supportClient.describeSeverityLevels(severityLevelsRequest);
        List<SeverityLevel> severityLevels = response.severityLevels();
        String levelName = null;
        for (SeverityLevel sevLevel : severityLevels) {
            System.out.println("The severity level name is: " +
sevLevel.name());
            if (sevLevel.name().compareTo("High") == 0)
                levelName = sevLevel.name();
        }
        return levelName;

    } catch (SupportException e) {
```

```
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
    return "";
}

// Return a List that contains a Service name and Category name.
public static List<String> displayServices(SupportClient supportClient) {
    try {
        DescribeServicesRequest servicesRequest =
DescribeServicesRequest.builder()
            .language("en")
            .build();

        DescribeServicesResponse response =
supportClient.describeServices(servicesRequest);
        String serviceCode = null;
        String catName = null;
        List<String> sevCatList = new ArrayList<>();
        List<Service> services = response.services();

        System.out.println("Get the first 10 services");
        int index = 1;
        for (Service service : services) {
            if (index == 11)
                break;

            System.out.println("The Service name is: " + service.name());
            if (service.name().compareTo("Account") == 0)
                serviceCode = service.code();

            // Get the Categories for this service.
            List<Category> categories = service.categories();
            for (Category cat : categories) {
                System.out.println("The category name is: " + cat.name());
                if (cat.name().compareTo("Security") == 0)
                    catName = cat.name();
            }
            index++;
        }

        // Push the two values to the list.
        sevCatList.add(serviceCode);
        sevCatList.add(catName);
    }
}
```

```
        return sevCatList;

    } catch (SupportException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
    return null;
}
}
```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella Documentazione di riferimento delle API AWS SDK for Java 2.x .
  - [AddAttachmentsToSet](#)
  - [AddCommunicationToCase](#)
  - [CreateCase](#)
  - [DescribeAttachment](#)
  - [DescribeCases](#)
  - [DescribeCommunications](#)
  - [DescribeServices](#)
  - [DescribeSeverityLevels](#)
  - [ResolveCase](#)

## Esempi di Systems Manager che utilizzano SDK for Java 2.x

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando AWS SDK for Java 2.x with Systems Manager.

Le operazioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le operazioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati e negli esempi tra servizi.

Scenari: esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio.

Ogni esempio include un collegamento a GitHub, dove è possibile trovare istruzioni su come configurare ed eseguire il codice nel contesto.

## Nozioni di base

### Salve Systems Manager

Il seguente esempio di codice mostra come iniziare a utilizzare Systems Manager.

#### SDK per Java 2.x

#### Note

C'è di più su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ssm.SsmClient;
import software.amazon.awssdk.services.ssm.model.DocumentFilter;
import software.amazon.awssdk.services.ssm.model.ListDocumentsRequest;
import software.amazon.awssdk.services.ssm.model.ListDocumentsResponse;

public class HelloSSM {

    public static void main(String[] args) {
        final String usage = ""

            Usage:
            <awsAccount>

            Where:
            awsAccount - Your AWS Account number.
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String awsAccount = args[0] ;
        Region region = Region.US_EAST_1;
        SsmClient ssmClient = SsmClient.builder()
            .region(region)
            .build();
```

```
        listDocuments(ssmClient, awsAccount);
    }

    /**
     * This code automatically fetches the next set of results using the `nextToken`
     * and
     * stops once the desired maxResults (20 in this case) have been reached.
     */
    public static void listDocuments(SsmClient ssmClient, String awsAccount) {
        String nextToken = null;
        int totalDocumentsReturned = 0;
        int maxResults = 20;
        do {
            ListDocumentsRequest request = ListDocumentsRequest.builder()
                .documentFilterList(
                    DocumentFilter.builder()
                        .key("Owner")
                        .value(awsAccount)
                        .build()
                )
                .maxResults(maxResults)
                .nextToken(nextToken)
                .build();

            ListDocumentsResponse response = ssmClient.listDocuments(request);
            response.documentIdentifiers().forEach(identifier ->
                System.out.println("Document Name: " + identifier.name()));
            nextToken = response.nextToken();
            totalDocumentsReturned += response.documentIdentifiers().size();
        } while (nextToken != null && totalDocumentsReturned < maxResults);
    }
}
```

- Per i dettagli sull'API, consulta [ListThings](#) in AWS SDK for Java 2.x API Reference.

## Argomenti

- [Azioni](#)
- [Scenari](#)

## Azioni

### CreateDocument

Il seguente esempio di codice mostra come utilizzare `CreateDocument`.

SDK per Java 2.x

#### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
// Create an AWS SSM document to use in this scenario.
public static void createSSMDoc(SsmClient ssmClient, String docName) {
    // Create JSON for the content
    String jsonData = ""
        {
            "schemaVersion": "2.2",
            "description": "Run a simple shell command",
            "mainSteps": [
                {
                    "action": "aws:runShellScript",
                    "name": "runEchoCommand",
                    "inputs": {
                        "runCommand": [
                            "echo 'Hello, world!'"
                        ]
                    }
                }
            ]
        }
        """;

    try {
        CreateDocumentRequest request = CreateDocumentRequest.builder()
            .content(jsonData)
            .name(docName)
            .documentType(DocumentType.COMMAND)
            .build();

        // Create the document.
    }
}
```



```
        CreateDocumentResponse response = ssmClient.createDocument(request);
        System.out.println("The status of the document is " +
response.documentDescription().status());

    } catch (DocumentAlreadyExistsException e) {
        System.err.println("The document already exists. Moving on." );
    } catch (SsmException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- Per i dettagli sull'API, [CreateDocument](#) consulta AWS SDK for Java 2.x API Reference.

## CreateMaintenanceWindow

Il seguente esempio di codice mostra come utilizzare `CreateMaintenanceWindow`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public static String createMaintenanceWindow(SsmClient ssmClient, String
winName) {
    CreateMaintenanceWindowRequest request =
CreateMaintenanceWindowRequest.builder()
        .name(winName)
        .description("This is my maintenance window")
        .allowUnassociatedTargets(true)
        .duration(2)
        .cutoff(1)
        .schedule("cron(0 10 ? * MON-FRI *)")
        .build();

    try {
        CreateMaintenanceWindowResponse response =
ssmClient.createMaintenanceWindow(request);
```

```
        String maintenanceWindowId = response.windowId();
        System.out.println("The maintenance window id is " +
maintenanceWindowId);
        return maintenanceWindowId;

    } catch (DocumentAlreadyExistsException e) {
        System.err.println("The maintenance window already exists. Moving on.");
    } catch (SsmException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }

    MaintenanceWindowFilter filter = MaintenanceWindowFilter.builder()
        .key("name")
        .values(winName)
        .build();

    DescribeMaintenanceWindowsRequest winRequest =
DescribeMaintenanceWindowsRequest.builder()
        .filters(filter)
        .build();

    String windowId = "";
    DescribeMaintenanceWindowsResponse response =
ssmClient.describeMaintenanceWindows(winRequest);
    List<MaintenanceWindowIdentity> windows = response.windowIdentities();
    if (!windows.isEmpty()) {
        windowId = windows.get(0).windowId();
        System.out.println("Window ID: " + windowId);
    } else {
        System.out.println("Window not found.");
    }
    return windowId;
}
```

- Per i dettagli sull'API, [CreateMaintenanceWindow](#) consulta AWS SDK for Java 2.x API Reference.

## CreateOpsItem

Il seguente esempio di codice mostra come utilizzare `CreateOpsItem`.

## SDK per Java 2.x

 Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
// Create an SSM OpsItem
public static String createSSMOpsItem(SsmClient ssmClient, String title, String
source, String category, String severity) {
    try {
        CreateOpsItemRequest opsItemRequest = CreateOpsItemRequest.builder()
            .description("Created by the Systems Manager Java API")
            .title(title)
            .source(source)
            .category(category)
            .severity(severity)
            .build();

        CreateOpsItemResponse itemResponse =
ssmClient.createOpsItem(opsItemRequest);
        return itemResponse.opsItemId();

    } catch (SsmException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return "";
}
```

- Per i dettagli sull'API, [CreateOpsItem](#) consulta AWS SDK for Java 2.x API Reference.

## DeleteDocument

Il seguente esempio di codice mostra come utilizzare `DeleteDocument`.

## SDK per Java 2.x

 Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
// Deletes an AWS Systems Manager document.
public static void deleteDoc(SsmClient ssmClient, String documentName) {
    try {
        DeleteDocumentRequest documentRequest = DeleteDocumentRequest.builder()
            .name(documentName)
            .build();


        ssmClient.deleteDocument(documentRequest);
        System.out.println("The Systems Manager document was successfully
deleted.");
    } catch (SsmException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- Per i dettagli sull'API, [DeleteDocument](#) consulta AWS SDK for Java 2.x API Reference.

## DeleteMaintenanceWindow

Il seguente esempio di codice mostra come utilizzare `DeleteMaintenanceWindow`.

## SDK per Java 2.x

 Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public static void deleteMaintenanceWindow(SsmClient ssmClient, String winId) {
    try {
        DeleteMaintenanceWindowRequest windowRequest =
DeleteMaintenanceWindowRequest.builder()
        .windowId(winId)
        .build();

        ssmClient.deleteMaintenanceWindow(windowRequest);
        System.out.println("The maintenance window was successfully deleted.");

    } catch (SsmException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- Per i dettagli sull'API, [DeleteMaintenanceWindow](#) consulta AWS SDK for Java 2.x API Reference.

## DescribeOpsItems

Il seguente esempio di codice mostra come utilizzare `DescribeOpsItems`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public static void describeOpsItems(SsmClient ssmClient, String key) {
    try {
        OpsItemFilter filter = OpsItemFilter.builder()
        .key(OpsItemFilterKey.OPS_ITEM_ID)
        .values(key)
        .operator(OpsItemFilterOperator.EQUAL)
        .build();

        DescribeOpsItemsRequest itemsRequest = DescribeOpsItemsRequest.builder()
```

```
        .maxResults(10)
        .opsItemFilters(filter)
        .build();

    DescribeOpsItemsResponse itemsResponse =
    ssmClient.describeOpsItems(itemsRequest);
    List<OpsItemSummary> items = itemsResponse.opsItemSummaries();
    for (OpsItemSummary item : items) {
        System.out.println("The item title is " + item.title() + " and the
status is "+item.status().toString());
    }

    } catch (SsmException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- Per i dettagli sull'API, [DescribeOpsItems](#) consulta AWS SDK for Java 2.x API Reference.

## DescribeParameters

Il seguente esempio di codice mostra come utilizzare `DescribeParameters`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ssm.SsmClient;
import software.amazon.awssdk.services.ssm.model.GetParameterRequest;
import software.amazon.awssdk.services.ssm.model.GetParameterResponse;
import software.amazon.awssdk.services.ssm.model.SsmException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 */
```

```
*
* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class GetParameter {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <paraName>

            Where:
                paraName - The name of the parameter.
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String paraName = args[0];
        Region region = Region.US_EAST_1;
        SsmClient ssmClient = SsmClient.builder()
            .region(region)
            .build();

        getParaValue(ssmClient, paraName);
        ssmClient.close();
    }

    public static void getParaValue(SsmClient ssmClient, String paraName) {
        try {
            GetParameterRequest parameterRequest = GetParameterRequest.builder()
                .name(paraName)
                .build();

            GetParameterResponse parameterResponse =
                ssmClient.getParameter(parameterRequest);
            System.out.println("The parameter value is " +
                parameterResponse.parameter().value());

        } catch (SsmException e) {
            System.err.println(e.getMessage());
        }
    }
}
```

```
        System.exit(1);
    }
}
}
```

- Per i dettagli sull'API, [DescribeParameters](#) consulta AWS SDK for Java 2.x API Reference.

## PutParameter

Il seguente esempio di codice mostra come utilizzare `PutParameter`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ssm.SsmClient;
import software.amazon.awssdk.services.ssm.model.ParameterType;
import software.amazon.awssdk.services.ssm.model.PutParameterRequest;
import software.amazon.awssdk.services.ssm.model.SsmException;

public class PutParameter {

    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <paraName>

            Where:
                paraName - The name of the parameter.
                paraValue - The value of the parameter.
            """;

        if (args.length != 2) {
            System.out.println(usage);
        }
    }
}
```



```
        System.exit(1);
    }

    String paraName = args[0];
    String paraValue = args[1];
    Region region = Region.US_EAST_1;
    SsmClient ssmClient = SsmClient.builder()
        .region(region)
        .build();

    putParaValue(ssmClient, paraName, paraValue);
    ssmClient.close();
}

public static void putParaValue(SsmClient ssmClient, String paraName, String
value) {
    try {
        PutParameterRequest parameterRequest = PutParameterRequest.builder()
            .name(paraName)
            .type(ParameterType.STRING)
            .value(value)
            .build();

        ssmClient.putParameter(parameterRequest);
        System.out.println("The parameter was successfully added.");

    } catch (SsmException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
}
```

- Per i dettagli sull'API, [PutParameter](#) consulta AWS SDK for Java 2.x API Reference.

## SendCommand

Il seguente esempio di codice mostra come utilizzare `SendCommand`.

## SDK per Java 2.x

 Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
// Sends a SSM command to a managed node.
public static String sendSSMCommand(SsmClient ssmClient, String documentName,
String instanceId) throws InterruptedException {
    // Before we use Document to send a command - make sure it is active.
    boolean isDocumentActive = false;
    DescribeDocumentRequest request = DescribeDocumentRequest.builder()
        .name(documentName)
        .build();

    while (!isDocumentActive) {
        DescribeDocumentResponse response = ssmClient.describeDocument(request);
        String documentStatus = response.document().statusAsString();
        if (documentStatus.equals("Active")) {
            System.out.println("The Systems Manager document is active and ready
to use.");
            isDocumentActive = true;
        } else {
            System.out.println("The Systems Manager document is not active.
Status: " + documentStatus);
            try {
                // Add a delay to avoid making too many requests.
                Thread.sleep(5000); // Wait for 5 seconds before checking again
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }

    // Create the SendCommandRequest.
    SendCommandRequest commandRequest = SendCommandRequest.builder()
        .documentName(documentName)
        .instanceIds(instanceId)
        .build();
}
```

```
// Send the command.
SendCommandResponse commandResponse = ssmClient.sendCommand(commandRequest);
String commandId = commandResponse.command().commandId();
System.out.println("The command Id is " + commandId);

// Wait for the command execution to complete.
GetCommandInvocationRequest invocationRequest =
GetCommandInvocationRequest.builder()
    .commandId(commandId)
    .instanceId(instanceId)
    .build();

System.out.println("Wait 5 secs");
TimeUnit.SECONDS.sleep(5);

// Retrieve the command execution details.
GetCommandInvocationResponse commandInvocationResponse =
ssmClient.getCommandInvocation(invocationRequest);

// Check the status of the command execution.
CommandInvocationStatus status = commandInvocationResponse.status();
if (status == CommandInvocationStatus.SUCCESS) {
    System.out.println("Command execution successful.");
} else {
    System.out.println("Command execution failed. Status: " + status);
}
return commandId;
}
```

- Per i dettagli sull'API, [SendCommand](#) consulta AWS SDK for Java 2.x API Reference.

## UpdateMaintenanceWindow

Il seguente esempio di codice mostra come utilizzare `UpdateMaintenanceWindow`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
// Update the maintenance window schedule
public static void updateSSMMaintenanceWindow(SsmClient ssmClient, String id,
String name) {
    try {
        UpdateMaintenanceWindowRequest updateRequest =
UpdateMaintenanceWindowRequest.builder()
            .windowId(id)
            .allowUnassociatedTargets(true)
            .duration(24)
            .enabled(true)
            .name(name)
            .schedule("cron(0 0 ? * MON *)")
            .build();

        ssmClient.updateMaintenanceWindow(updateRequest);
        System.out.println("The Systems Manager maintenance window was
successfully updated.");

    } catch (SsmException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- Per i dettagli sull'API, [UpdateMaintenanceWindow](#) consulta AWS SDK for Java 2.x API Reference.

## UpdateOpsItem

Il seguente esempio di codice mostra come utilizzare `UpdateOpsItem`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public static void resolveOpsItem(SsmClient ssmClient, String opsID) {
```

```
try {
    UpdateOpsItemRequest opsItemRequest = UpdateOpsItemRequest.builder()
        .opsItemId(opsID)
        .status(OpsItemStatus.RESOLVED)
        .build();

    ssmClient.updateOpsItem(opsItemRequest);

} catch (SsmException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
}
```

- Per i dettagli sull'API, [UpdateOpsItem](#) consulta AWS SDK for Java 2.x API Reference.

## Scenari

### Inizia a usare Systems Manager

L'esempio di codice seguente mostra come utilizzare le finestre, i documenti e di manutenzione di Systems Manager OpsItems.

### SDK per Java 2.x

#### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ssm.SsmClient;
import software.amazon.awssdk.services.ssm.model.CommandInvocation;
import software.amazon.awssdk.services.ssm.model.CommandInvocationStatus;
import software.amazon.awssdk.services.ssm.model.CreateDocumentRequest;
import software.amazon.awssdk.services.ssm.model.CreateDocumentResponse;
import software.amazon.awssdk.services.ssm.model.CreateMaintenanceWindowRequest;
import software.amazon.awssdk.services.ssm.model.CreateMaintenanceWindowResponse;
import software.amazon.awssdk.services.ssm.model.CreateOpsItemRequest;
```

```
import software.amazon.awssdk.services.ssm.model.CreateOpsItemResponse;
import software.amazon.awssdk.services.ssm.model.DeleteDocumentRequest;
import software.amazon.awssdk.services.ssm.model.DeleteMaintenanceWindowRequest;
import software.amazon.awssdk.services.ssm.model.DeleteOpsItemRequest;
import software.amazon.awssdk.services.ssm.model.DescribeDocumentRequest;
import software.amazon.awssdk.services.ssm.model.DescribeDocumentResponse;
import software.amazon.awssdk.services.ssm.model.DescribeMaintenanceWindowsRequest;
import software.amazon.awssdk.services.ssm.model.DescribeMaintenanceWindowsResponse;
import software.amazon.awssdk.services.ssm.model.DescribeOpsItemsRequest;
import software.amazon.awssdk.services.ssm.model.DescribeOpsItemsResponse;
import software.amazon.awssdk.services.ssm.model.DocumentAlreadyExistsException;
import software.amazon.awssdk.services.ssm.model.DocumentType;
import software.amazon.awssdk.services.ssm.model.GetCommandInvocationRequest;
import software.amazon.awssdk.services.ssm.model.GetCommandInvocationResponse;
import software.amazon.awssdk.services.ssm.model.GetOpsItemRequest;
import software.amazon.awssdk.services.ssm.model.GetOpsItemResponse;
import software.amazon.awssdk.services.ssm.model.ListCommandInvocationsRequest;
import software.amazon.awssdk.services.ssm.model.ListCommandInvocationsResponse;
import software.amazon.awssdk.services.ssm.model.MaintenanceWindowFilter;
import software.amazon.awssdk.services.ssm.model.MaintenanceWindowIdentity;
import software.amazon.awssdk.services.ssm.model.OpsItemDataValue;
import software.amazon.awssdk.services.ssm.model.OpsItemFilter;
import software.amazon.awssdk.services.ssm.model.OpsItemFilterKey;
import software.amazon.awssdk.services.ssm.model.OpsItemFilterOperator;
import software.amazon.awssdk.services.ssm.model.OpsItemStatus;
import software.amazon.awssdk.services.ssm.model.OpsItemSummary;
import software.amazon.awssdk.services.ssm.model.SendCommandRequest;
import software.amazon.awssdk.services.ssm.model.SendCommandResponse;
import software.amazon.awssdk.services.ssm.model.SsmException;
import software.amazon.awssdk.services.ssm.model.UpdateMaintenanceWindowRequest;
import software.amazon.awssdk.services.ssm.model.UpdateOpsItemRequest;
import java.time.ZoneId;
import java.time.format.DateTimeFormatter;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.Scanner;
import java.util.concurrent.TimeUnit;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
```

```

*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/setup.html
*
*
* This Java program performs these tasks:
* 1. Creates an AWS Systems Manager maintenance window with a default name or a
user-provided name.
* 2. Modifies the maintenance window schedule.
* 3. Creates a Systems Manager document with a default name or a user-provided
name.
* 4. Sends a command to a specified EC2 instance using the created Systems Manager
document and displays the time when the command was invoked.
* 5. Creates a Systems Manager OpsItem with a predefined title, source, category,
and severity.
* 6. Updates and resolves the created OpsItem.
* 7. Deletes the Systems Manager maintenance window, OpsItem, and document.
*/

public class SSMSscenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");
    public static void main(String[] args) throws InterruptedException {
        String usage = ""
            Usage:
                <instanceId> <title> <source> <category> <severity>

        Where:
            instanceId - The Amazon EC2 Linux/UNIX instance Id that AWS Systems
Manager uses (ie, i-0149338494ed95f06).
            title - The title of the parameter (default is Disk Space Alert).
            source - The source of the parameter (default is EC2).
            category - The category of the parameter. Valid values are
'Availability', 'Cost', 'Performance', 'Recovery', 'Security' (default is
Performance).
            severity - The severity of the parameter. Severity should be a
number from 1 to 4 (default is 2).
        """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        Scanner scanner = new Scanner(System.in);
        String documentName;

```

```
String windowName;
String instanceId = args[0];
String title = "Disk Space Alert" ;
String source = "EC2" ;
String category = "Performance" ;
String severity = "2" ;

Region region = Region.US_EAST_1;
SsmClient ssmClient = SsmClient.builder()
    .region(region)
    .build();

System.out.println(DASHES);
System.out.println("""
    Welcome to the AWS Systems Manager SDK Getting Started scenario.
    This program demonstrates how to interact with Systems Manager using the
AWS SDK for Java (v2).
    Systems Manager is the operations hub for your AWS applications and
resources and a secure end-to-end management solution.
    The program's primary functions include creating a maintenance window,
creating a document, sending a command to a document,
    listing documents, listing commands, creating an OpsItem, modifying an
OpsItem, and deleting Systems Manager resources.
    Upon completion of the program, all AWS resources are cleaned up.
    Let's get started...
    Please hit Enter
    """);
scanner.nextLine();
System.out.println(DASHES);

System.out.println("Create a Systems Manager maintenance window.");
System.out.println("Please enter the maintenance window name (default is
ssm-maintenance-window):");
String win = scanner.nextLine();
windowName = win.isEmpty() ? "ssm-maintenance-window" : win;
String winId = createMaintenanceWindow(ssmClient, windowName);
System.out.println(DASHES);

System.out.println("Modify the maintenance window by changing the
schedule");
System.out.println("Please hit Enter");
scanner.nextLine();
updateSSMMaintenanceWindow(ssmClient, winId, windowName);
System.out.println(DASHES);
```



```
System.out.println("Create a document that defines the actions that Systems
Manager performs on your EC2 instance.");
System.out.println("Please enter the document name (default is
ssmdocument):");
String doc = scanner.nextLine();
documentName = doc.isEmpty() ? "ssmdocument" : doc;
createSSMDoc(ssmClient, documentName);

System.out.println("Now we are going to run a command on an EC2 instance
that echoes 'Hello, world!'");
System.out.println("Please hit Enter");
scanner.nextLine();
String commandId = sendSSMCommand(ssmClient, documentName, instanceId);
System.out.println(DASHES);

System.out.println("Lets get the time when the specific command was sent to
the specific managed node");
System.out.println("Please hit Enter");
scanner.nextLine();
displayCommands(ssmClient, commandId);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("""
    Now we will create a Systems Manager OpsItem.
    An OpsItem is a feature provided by the Systems Manager service.
    It is a type of operational data item that allows you to manage and
track various operational issues,
    events, or tasks within your AWS environment.

    You can create OpsItems to track and manage operational issues as they
arise.

    For example, you could create an OpsItem whenever your application
detects a critical error
    or an anomaly in your infrastructure.
    """);

System.out.println("Please hit Enter");
scanner.nextLine();
String opsItemId = createSSMOpsItem(ssmClient, title, source, category,
severity);
System.out.println(DASHES);
```

```
System.out.println(DASHES);
System.out.println("Now we will update the OpsItem "+opsItemId);
System.out.println("Please hit Enter");
scanner.nextLine();
String description = "An update to "+opsItemId ;
updateOpsItem(ssmClient, opsItemId, title, description);
System.out.println("Now we will get the status of the OpsItem "+opsItemId);
System.out.println("Please hit Enter");
scanner.nextLine();
describeOpsItems(ssmClient, opsItemId);
System.out.println("Now we will resolve the OpsItem "+opsItemId);
System.out.println("Please hit Enter");
scanner.nextLine();
resolveOpsItem(ssmClient, opsItemId);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("Would you like to delete the Systems Manager resources?
(y/n)");
String delAns = scanner.nextLine().trim();
if (delAns.equalsIgnoreCase("y")) {
    System.out.println("You selected to delete the resources.");
    System.out.print("Press Enter to continue...");
    scanner.nextLine();
    deleteOpsItem(ssmClient, opsItemId);
    deleteMaintenanceWindow(ssmClient, winId);
    deleteDoc(ssmClient, documentName);
} else {
    System.out.println("The Systems Manager resources will not be deleted");
}
System.out.println(DASHES);

System.out.println("This concludes the Systems Manager SDK Getting Started
scenario.");
System.out.println(DASHES);
}

// Displays the date and time when the specific command was invoked.
public static void displayCommands(SsmClient ssmClient, String commandId) {
    try {
        ListCommandInvocationsRequest commandInvocationsRequest =
ListCommandInvocationsRequest.builder()
        .commandId(commandId)
        .build();
```

```
        ListCommandInvocationsResponse response =
ssmClient.listCommandInvocations(commandInvocationsRequest);
        List<CommandInvocation> commandList = response.commandInvocations();
        DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd
HH:mm:ss").withZone(ZoneId.systemDefault());
        for (CommandInvocation invocation : commandList) {
            System.out.println("The time of the command invocation is " +
formatter.format(invocation.requestedDateTime()));
        }

    } catch (SsmException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

// Create an SSM OpsItem
public static String createSSMOpsItem(SsmClient ssmClient, String title, String
source, String category, String severity) {
    try {
        CreateOpsItemRequest opsItemRequest = CreateOpsItemRequest.builder()
            .description("Created by the Systems Manager Java API")
            .title(title)
            .source(source)
            .category(category)
            .severity(severity)
            .build();

        CreateOpsItemResponse itemResponse =
ssmClient.createOpsItem(opsItemRequest);
        return itemResponse.opsItemId();

    } catch (SsmException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return "";
}

// Update the AWS SSM OpsItem.
public static void updateOpsItem(SsmClient ssmClient, String opsItemId, String
title, String description) {
    Map<String, OpsItemDataValue> operationalData = new HashMap<>();
```

```
        operationalData.put("key1",
OpsItemDataValue.builder().value("value1").build());
        operationalData.put("key2",
OpsItemDataValue.builder().value("value2").build());

    try {
        UpdateOpsItemRequest request = UpdateOpsItemRequest.builder()
            .opsItemId(opsItemId)
            .title(title)
            .operationalData(operationalData)
            .status(getOpsItem(ssmClient, opsItemId))
            .description(description)
            .build();

        ssmClient.updateOpsItem(request);

    } catch (SsmException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void resolveOpsItem(SsmClient ssmClient, String opsID) {
    try {
        UpdateOpsItemRequest opsItemRequest = UpdateOpsItemRequest.builder()
            .opsItemId(opsID)
            .status(OpsItemStatus.RESOLVED)
            .build();

        ssmClient.updateOpsItem(opsItemRequest);

    } catch (SsmException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

// Gets a specific OpsItem.
private static OpsItemStatus getOpsItem(SsmClient ssmClient, String opsItemId) {
    GetOpsItemRequest itemRequest = GetOpsItemRequest.builder()
        .opsItemId(opsItemId)
        .build();

    try {
```

```
        GetOpsItemResponse response = ssmClient.getOpsItem(itemRequest);
        return response.opsItem().status();

    } catch (SsmException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return null;
}

// Sends a SSM command to a managed node.
public static String sendSSMCommand(SsmClient ssmClient, String documentName,
String instanceId) throws InterruptedException {
    // Before we use Document to send a command - make sure it is active.
    boolean isDocumentActive = false;
    DescribeDocumentRequest request = DescribeDocumentRequest.builder()
        .name(documentName)
        .build();

    while (!isDocumentActive) {
        DescribeDocumentResponse response = ssmClient.describeDocument(request);
        String documentStatus = response.document().statusAsString();
        if (documentStatus.equals("Active")) {
            System.out.println("The Systems Manager document is active and ready
to use.");
            isDocumentActive = true;
        } else {
            System.out.println("The Systems Manager document is not active.
Status: " + documentStatus);
            try {
                // Add a delay to avoid making too many requests.
                Thread.sleep(5000); // Wait for 5 seconds before checking again
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }

    // Create the SendCommandRequest.
    SendCommandRequest commandRequest = SendCommandRequest.builder()
        .documentName(documentName)
        .instanceIds(instanceId)
        .build();
}
```

```
// Send the command.
SendCommandResponse commandResponse = ssmClient.sendCommand(commandRequest);
String commandId = commandResponse.command().commandId();
System.out.println("The command Id is " + commandId);

// Wait for the command execution to complete.
GetCommandInvocationRequest invocationRequest =
GetCommandInvocationRequest.builder()
    .commandId(commandId)
    .instanceId(instanceId)
    .build();

System.out.println("Wait 5 secs");
TimeUnit.SECONDS.sleep(5);

// Retrieve the command execution details.
GetCommandInvocationResponse commandInvocationResponse =
ssmClient.getCommandInvocation(invocationRequest);

// Check the status of the command execution.
CommandInvocationStatus status = commandInvocationResponse.status();
if (status == CommandInvocationStatus.SUCCESS) {
    System.out.println("Command execution successful.");
} else {
    System.out.println("Command execution failed. Status: " + status);
}
return commandId;
}

// Deletes an AWS Systems Manager document.
public static void deleteDoc(SsmClient ssmClient, String documentName) {
    try {
        DeleteDocumentRequest documentRequest = DeleteDocumentRequest.builder()
            .name(documentName)
            .build();

        ssmClient.deleteDocument(documentRequest);
        System.out.println("The Systems Manager document was successfully
deleted.");

    } catch (SsmException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

```
}

public static void deleteMaintenanceWindow(SsmClient ssmClient, String winId) {
    try {
        DeleteMaintenanceWindowRequest windowRequest =
DeleteMaintenanceWindowRequest.builder()
            .windowId(winId)
            .build();

        ssmClient.deleteMaintenanceWindow(windowRequest);
        System.out.println("The maintenance window was successfully deleted.");

    } catch (SsmException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

// Update the maintenance window schedule
public static void updateSSMMaintenanceWindow(SsmClient ssmClient, String id,
String name) {
    try {
        UpdateMaintenanceWindowRequest updateRequest =
UpdateMaintenanceWindowRequest.builder()
            .windowId(id)
            .allowUnassociatedTargets(true)
            .duration(24)
            .enabled(true)
            .name(name)
            .schedule("cron(0 0 ? * MON *)")
            .build();

        ssmClient.updateMaintenanceWindow(updateRequest);
        System.out.println("The Systems Manager maintenance window was
successfully updated.");

    } catch (SsmException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static String createMaintenanceWindow(SsmClient ssmClient, String
winName) {
```

```
    CreateMaintenanceWindowRequest request =
CreateMaintenanceWindowRequest.builder()
    .name(winName)
    .description("This is my maintenance window")
    .allowUnassociatedTargets(true)
    .duration(2)
    .cutoff(1)
    .schedule("cron(0 10 ? * MON-FRI *)")
    .build();

    try {
        CreateMaintenanceWindowResponse response =
ssmClient.createMaintenanceWindow(request);
        String maintenanceWindowId = response.windowId();
        System.out.println("The maintenance window id is " +
maintenanceWindowId);
        return maintenanceWindowId;

    } catch (DocumentAlreadyExistsException e) {
        System.err.println("The maintenance window already exists. Moving on.");
    } catch (SsmException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }

    MaintenanceWindowFilter filter = MaintenanceWindowFilter.builder()
    .key("name")
    .values(winName)
    .build();

    DescribeMaintenanceWindowsRequest winRequest =
DescribeMaintenanceWindowsRequest.builder()
    .filters(filter)
    .build();

    String windowId = "";
    DescribeMaintenanceWindowsResponse response =
ssmClient.describeMaintenanceWindows(winRequest);
    List<MaintenanceWindowIdentity> windows = response.windowIdentities();
    if (!windows.isEmpty()) {
        windowId = windows.get(0).windowId();
        System.out.println("Window ID: " + windowId);
    } else {
        System.out.println("Window not found.");
    }
}
```



```
    }
    return windowId;
}

// Create an AWS SSM document to use in this scenario.
public static void createSSMDoc(SsmClient ssmClient, String docName) {
    // Create JSON for the content
    String jsonData = ""
        {
            "schemaVersion": "2.2",
            "description": "Run a simple shell command",
            "mainSteps": [
                {
                    "action": "aws:runShellScript",
                    "name": "runEchoCommand",
                    "inputs": {
                        "runCommand": [
                            "echo 'Hello, world!'"
                        ]
                    }
                }
            ]
        }
        """;

    try {
        CreateDocumentRequest request = CreateDocumentRequest.builder()
            .content(jsonData)
            .name(docName)
            .documentType(DocumentType.COMMAND)
            .build();

        // Create the document.
        CreateDocumentResponse response = ssmClient.createDocument(request);
        System.out.println("The status of the document is " +
response.documentDescription().status());

        } catch (DocumentAlreadyExistsException e) {
            System.err.println("The document already exists. Moving on." );
        } catch (SsmException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

```
public static void describeOpsItems(SsmClient ssmClient, String key) {
    try {
        OpsItemFilter filter = OpsItemFilter.builder()
            .key(OpsItemFilterKey.OPS_ITEM_ID)
            .values(key)
            .operator(OpsItemFilterOperator.EQUAL)
            .build();

        DescribeOpsItemsRequest itemsRequest = DescribeOpsItemsRequest.builder()
            .maxResults(10)
            .opsItemFilters(filter)
            .build();

        DescribeOpsItemsResponse itemsResponse =
ssmClient.describeOpsItems(itemsRequest);
        List<OpsItemSummary> items = itemsResponse.opsItemSummaries();
        for (OpsItemSummary item : items) {
            System.out.println("The item title is " + item.title() + " and the
status is "+item.status().toString());
        }

    } catch (SsmException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void deleteOpsItem(SsmClient ssmClient, String opsId) {
    try {
        DeleteOpsItemRequest deleteOpsItemRequest =
DeleteOpsItemRequest.builder()
            .opsItemId(opsId)
            .build();

        ssmClient.deleteOpsItem(deleteOpsItemRequest);
        System.out.println(opsId + " Opsitem was deleted");

    } catch (SsmException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
}
```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella Documentazione di riferimento delle API AWS SDK for Java 2.x .
  - [CommandInvocations](#)
  - [CreateDocument](#)
  - [CreateMaintenanceWindow](#)
  - [CreateOpsItem](#)
  - [DeleteMaintenanceWindow](#)
  - [SendCommand](#)
  - [UpdateOpsItem](#)

## Esempi di Amazon Textract con SDK for Java 2.x

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando Amazon Textract. AWS SDK for Java 2.x

Le operazioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le operazioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati e negli esempi tra servizi.

Scenari: esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio.

Ogni esempio include un collegamento a GitHub, dove puoi trovare istruzioni su come configurare ed eseguire il codice nel contesto.

Argomenti

- [Azioni](#)

Azioni

### **AnalyzeDocument**

Il seguente esempio di codice mostra come utilizzare `AnalyzeDocument`.

## SDK per Java 2.x

 Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.textract.TextractClient;
import software.amazon.awssdk.services.textract.model.AnalyzeDocumentRequest;
import software.amazon.awssdk.services.textract.model.Document;
import software.amazon.awssdk.services.textract.model.FeatureType;
import software.amazon.awssdk.services.textract.model.AnalyzeDocumentResponse;
import software.amazon.awssdk.services.textract.model.Block;
import software.amazon.awssdk.services.textract.model.TextractException;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class AnalyzeDocument {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <sourceDoc>\s

                Where:
```

```
        sourceDoc - The path where the document is located (must be an
image, for example, C:/AWS/book.png).\s
        """";

    if (args.length != 1) {
        System.out.println(usage);
        System.exit(1);
    }

    String sourceDoc = args[0];
    Region region = Region.US_EAST_2;
    TextractClient textractClient = TextractClient.builder()
        .region(region)
        .build();

    analyzeDoc(textractClient, sourceDoc);
    textractClient.close();
}

public static void analyzeDoc(TextractClient textractClient, String sourceDoc) {
    try {
        InputStream sourceStream = new FileInputStream(new File(sourceDoc));
        SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);

        // Get the input Document object as bytes
        Document myDoc = Document.builder()
            .bytes(sourceBytes)
            .build();

        List<FeatureType> featureTypes = new ArrayList<FeatureType>();
        featureTypes.add(FeatureType.FORMS);
        featureTypes.add(FeatureType.TABLES);

        AnalyzeDocumentRequest analyzeDocumentRequest =
    AnalyzeDocumentRequest.builder()
        .featureTypes(featureTypes)
        .document(myDoc)
        .build();

        AnalyzeDocumentResponse analyzeDocument =
    textractClient.analyzeDocument(analyzeDocumentRequest);
        List<Block> docInfo = analyzeDocument.blocks();
        Iterator<Block> blockIterator = docInfo.iterator();
    }
```

```
        while (blockIterator.hasNext()) {
            Block block = blockIterator.next();
            System.out.println("The block type is " +
block.blockType().toString());
        }

    } catch (TextractException | FileNotFoundException e) {

        System.err.println(e.getMessage());
        System.exit(1);
    }
}
}
```

- Per i dettagli sull'API, [AnalyzeDocument](#) consulta AWS SDK for Java 2.x API Reference.

## DetectDocumentText

Il seguente esempio di codice mostra come utilizzare `DetectDocumentText`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Rileva il testo da un documento di input.

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.textract.TextractClient;
import software.amazon.awssdk.services.textract.model.Document;
import software.amazon.awssdk.services.textract.model.DetectDocumentTextRequest;
import software.amazon.awssdk.services.textract.model.DetectDocumentTextResponse;
import software.amazon.awssdk.services.textract.model.Block;
import software.amazon.awssdk.services.textract.model.DocumentMetadata;
import software.amazon.awssdk.services.textract.model.TextractException;
import java.io.File;
import java.io.FileInputStream;
```

```
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DetectDocumentText {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <sourceDoc>\s

                Where:
                sourceDoc - The path where the document is located (must be an
image, for example, C:/AWS/book.png).\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String sourceDoc = args[0];
        Region region = Region.US_EAST_2;
        TextractClient textractClient = TextractClient.builder()
                .region(region)
                .build();

        detectDocText(textractClient, sourceDoc);
        textractClient.close();
    }

    public static void detectDocText(TextractClient textractClient, String
sourceDoc) {
        try {
            InputStream sourceStream = new FileInputStream(new File(sourceDoc));
            SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);
```

```
        // Get the input Document object as bytes.
        Document myDoc = Document.builder()
            .bytes(sourceBytes)
            .build();

        DetectDocumentTextRequest detectDocumentTextRequest =
DetectDocumentTextRequest.builder()
            .document(myDoc)
            .build();

        // Invoke the Detect operation.
        DetectDocumentTextResponse textResponse =
textextractClient.detectDocumentText(detectDocumentTextRequest);
        List<Block> docInfo = textResponse.blocks();
        for (Block block : docInfo) {
            System.out.println("The block type is " +
block.blockType().toString());
        }

        DocumentMetadata documentMetadata = textResponse.documentMetadata();
        System.out.println("The number of pages in the document is " +
documentMetadata.pages());

    } catch (TextextractException | FileNotFoundException e) {

        System.err.println(e.getMessage());
        System.exit(1);
    }
}
}
```

Rileva il testo da un documento che si trova in un bucket Amazon S3.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.textextract.model.S3Object;
import software.amazon.awssdk.services.textextract.TextextractClient;
import software.amazon.awssdk.services.textextract.model.Document;
import software.amazon.awssdk.services.textextract.model.DetectDocumentTextRequest;
import software.amazon.awssdk.services.textextract.model.DetectDocumentTextResponse;
import software.amazon.awssdk.services.textextract.model.Block;
import software.amazon.awssdk.services.textextract.model.DocumentMetadata;
```



```
import software.amazon.awssdk.services.textract.model.TextractException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DetectDocumentTextS3 {

    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <bucketName> <docName>\s

            Where:
                bucketName - The name of the Amazon S3 bucket that contains the
document.\s

                docName - The document name (must be an image, i.e., book.png).
\s

            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String docName = args[1];
        Region region = Region.US_WEST_2;
        TextractClient textractClient = TextractClient.builder()
            .region(region)
            .build();

        detectDocTextS3(textractClient, bucketName, docName);
        textractClient.close();
    }

    public static void detectDocTextS3(TextractClient textractClient, String
bucketName, String docName) {
```

```
try {
    S3Object s3Object = S3Object.builder()
        .bucket(bucketName)
        .name(docName)
        .build();

    // Create a Document object and reference the s3Object instance.
    Document myDoc = Document.builder()
        .s3Object(s3Object)
        .build();

    DetectDocumentTextRequest detectDocumentTextRequest =
DetectDocumentTextRequest.builder()
        .document(myDoc)
        .build();

    DetectDocumentTextResponse textResponse =
textextractClient.detectDocumentText(detectDocumentTextRequest);
    for (Block block : textResponse.blocks()) {
        System.out.println("The block type is " +
block.blockType().toString());
    }

    DocumentMetadata documentMetadata = textResponse.documentMetadata();
    System.out.println("The number of pages in the document is " +
documentMetadata.pages());

} catch (TextractException e) {

    System.err.println(e.getMessage());
    System.exit(1);
}
}
```

- Per i dettagli sull'API, consulta la sezione [AWS SDK for Java 2.x API DetectDocumentTextReference](#).

## StartDocumentAnalysis

Il seguente esempio di codice mostra come utilizzare `StartDocumentAnalysis`.

## SDK per Java 2.x

 Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.textract.model.S3Object;
import software.amazon.awssdk.services.textract.TextractClient;
import software.amazon.awssdk.services.textract.model.StartDocumentAnalysisRequest;
import software.amazon.awssdk.services.textract.model.DocumentLocation;
import software.amazon.awssdk.services.textract.model.TextractException;
import software.amazon.awssdk.services.textract.model.StartDocumentAnalysisResponse;
import software.amazon.awssdk.services.textract.model.GetDocumentAnalysisRequest;
import software.amazon.awssdk.services.textract.model.GetDocumentAnalysisResponse;
import software.amazon.awssdk.services.textract.model.FeatureType;
import java.util.ArrayList;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class StartDocumentAnalysis {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <bucketName> <docName>\s

                Where:
                bucketName - The name of the Amazon S3 bucket that contains the
document.\s
                docName - The document name (must be an image, for example,
book.png).\s

                """;
```

```
    if (args.length != 2) {
        System.out.println(usage);
        System.exit(1);
    }

    String bucketName = args[0];
    String docName = args[1];
    Region region = Region.US_WEST_2;
    TextractClient textractClient = TextractClient.builder()
        .region(region)
        .build();

    String jobId = startDocAnalysisS3(textractClient, bucketName, docName);
    System.out.println("Getting results for job " + jobId);
    String status = getJobResults(textractClient, jobId);
    System.out.println("The job status is " + status);
    textractClient.close();
}

public static String startDocAnalysisS3(TextractClient textractClient, String
bucketName, String docName) {
    try {
        List<FeatureType> myList = new ArrayList<>();
        myList.add(FeatureType.TABLES);
        myList.add(FeatureType.FORMS);

        S3Object s3Object = S3Object.builder()
            .bucket(bucketName)
            .name(docName)
            .build();

        DocumentLocation location = DocumentLocation.builder()
            .s3Object(s3Object)
            .build();

        StartDocumentAnalysisRequest documentAnalysisRequest =
StartDocumentAnalysisRequest.builder()
            .documentLocation(location)
            .featureTypes(myList)
            .build();

        StartDocumentAnalysisResponse response =
textractClient.startDocumentAnalysis(documentAnalysisRequest);
```

```
        // Get the job ID
        String jobId = response.jobId();
        return jobId;

    } catch (TexttractException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return "";
}

private static String getJobResults(TexttractClient texttractClient, String jobId)
{
    boolean finished = false;
    int index = 0;
    String status = "";

    try {
        while (!finished) {
            GetDocumentAnalysisRequest analysisRequest =
GetDocumentAnalysisRequest.builder()
                .jobId(jobId)
                .maxResults(1000)
                .build();

            GetDocumentAnalysisResponse response =
texttractClient.getDocumentAnalysis(analysisRequest);
            status = response.jobStatus().toString();

            if (status.compareTo("SUCCEEDED") == 0)
                finished = true;
            else {
                System.out.println(index + " status is: " + status);
                Thread.sleep(1000);
            }
            index++;
        }

        return status;
    } catch (InterruptedException e) {
        System.out.println(e.getMessage());
        System.exit(1);
    }
}
```

```
    }  
    return "";  
  }  
}
```

- Per i dettagli sull'API, [StartDocumentAnalysis](#) consulta AWS SDK for Java 2.x API Reference.

## Esempi di Amazon Transcribe con SDK for Java 2.x

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando Amazon Transcribe. AWS SDK for Java 2.x

Le operazioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le operazioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati e negli esempi tra servizi.

Scenari: esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio.

Ogni esempio include un collegamento a GitHub, dove puoi trovare istruzioni su come configurare ed eseguire il codice nel contesto.

### Argomenti

- [Azioni](#)
- [Scenari](#)

### Azioni

#### **ListTranscriptionJobs**

Il seguente esempio di codice mostra come utilizzare `ListTranscriptionJobs`.

SDK per Java 2.x

#### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public class ListTranscriptionJobs {
    public static void main(String[] args) {
        TranscribeClient transcribeClient = TranscribeClient.builder()
            .region(Region.US_EAST_1)
            .build();

        listTranscriptionJobs(transcribeClient);
    }

    public static void listTranscriptionJobs(TranscribeClient transcribeClient)
    {
        ListTranscriptionJobsRequest listJobsRequest =
        ListTranscriptionJobsRequest.builder()
            .build();

        transcribeClient.listTranscriptionJobsPaginator(listJobsRequest).stream()
            .flatMap(response -> response.transcriptionJobSummaries().stream())
            .forEach(jobSummary -> {
                System.out.println("Job Name: " +
                jobSummary.transcriptionJobName());
                System.out.println("Job Status: " +
                jobSummary.transcriptionJobStatus());
                System.out.println("Output Location: " +
                jobSummary.outputLocationType());
                // Add more information as needed

                // Retrieve additional details for the job if necessary
                GetTranscriptionJobResponse jobDetails =
                transcribeClient.getTranscriptionJob(
                    GetTranscriptionJobRequest.builder()
                        .transcriptionJobName(jobSummary.transcriptionJobName())
                        .build());

                // Display additional details
                System.out.println("Language Code: " +
                jobDetails.transcriptionJob().languageCode());
                System.out.println("Media Format: " +
                jobDetails.transcriptionJob().mediaFormat());
                // Add more details as needed

                System.out.println("-----");
            });
    }
}
```

```
    }  
}
```

- Per i dettagli sull'API, [ListTranscriptionJobs](#) consulta AWS SDK for Java 2.x API Reference.

## StartTranscriptionJob

Il seguente esempio di codice mostra come utilizzare `StartTranscriptionJob`.

SDK per Java 2.x

### Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public class TranscribeStreamingDemoApp {  
    private static final Region REGION = Region.US_EAST_1;  
    private static TranscribeStreamingAsyncClient client;  
  
    public static void main(String args[])  
        throws URISyntaxException, ExecutionException, InterruptedException,  
        LineUnavailableException {  
  
        client = TranscribeStreamingAsyncClient.builder()  
            .credentialsProvider(getCredentials())  
            .region(REGION)  
            .build();  
  
        CompletableFuture<Void> result =  
client.startStreamTranscription(getRequest(16_000),  
    new AudioStreamPublisher(getStreamFromMic()),  
    getResponseHandler());  
  
        result.get();  
        client.close();  
    }  
  
    private static InputStream getStreamFromMic() throws LineUnavailableException {
```



```
// Signed PCM AudioFormat with 16kHz, 16 bit sample size, mono
int sampleRate = 16000;
AudioFormat format = new AudioFormat(sampleRate, 16, 1, true, false);
DataLine.Info info = new DataLine.Info(TargetDataLine.class, format);

if (!AudioSystem.isLineSupported(info)) {
    System.out.println("Line not supported");
    System.exit(0);
}

TargetDataLine line = (TargetDataLine) AudioSystem.getLine(info);
line.open(format);
line.start();

InputStream audioStream = new AudioInputStream(line);
return audioStream;
}

private static AwsCredentialsProvider getCredentials() {
    return DefaultCredentialsProvider.create();
}

private static StartStreamTranscriptionRequest getRequest(Integer
mediaSampleRateHertz) {
    return StartStreamTranscriptionRequest.builder()
        .languageCode(LanguageCode.EN_US.toString())
        .mediaEncoding(MediaEncoding.PCM)
        .mediaSampleRateHertz(mediaSampleRateHertz)
        .build();
}

private static StartStreamTranscriptionResponseHandler getResponseHandler() {
    return StartStreamTranscriptionResponseHandler.builder()
        .onResponse(r -> {
            System.out.println("Received Initial response");
        })
        .onError(e -> {
            System.out.println(e.getMessage());
            StringWriter sw = new StringWriter();
            e.printStackTrace(new PrintWriter(sw));
            System.out.println("Error Occurred: " + sw.toString());
        })
        .onComplete(() -> {
            System.out.println("=== All records stream successfully ===");
        });
}
```

```

        })
        .subscriber(event -> {
            List<Result> results = ((TranscriptEvent)
event).transcript().results();
            if (results.size() > 0) {
                if (!
results.get(0).alternatives().get(0).transcript().isEmpty()) {

System.out.println(results.get(0).alternatives().get(0).transcript());
                }
            }
        })
        .build();
    }

    private InputStream getStreamFromFile(String audioFileName) {
        try {
            File inputFile = new
File(getClass().getClassLoader().getResource(audioFileName).getFile());
            InputStream audioStream = new FileInputStream(inputFile);
            return audioStream;
        } catch (FileNotFoundException e) {
            throw new RuntimeException(e);
        }
    }

    private static class AudioStreamPublisher implements Publisher<AudioStream> {
        private final InputStream inputStream;
        private static Subscription currentSubscription;

        private AudioStreamPublisher(InputStream inputStream) {
            this.inputStream = inputStream;
        }

        @Override
        public void subscribe(Subscriber<? super AudioStream> s) {

            if (this.currentSubscription == null) {
                this.currentSubscription = new SubscriptionImpl(s, inputStream);
            } else {
                this.currentSubscription.cancel();
                this.currentSubscription = new SubscriptionImpl(s, inputStream);
            }
            s.onSubscribe(currentSubscription);
        }
    }

```

```
    }  
  }  
  
  public static class SubscriptionImpl implements Subscription {  
    private static final int CHUNK_SIZE_IN_BYTES = 1024 * 1;  
    private final Subscriber<? super AudioStream> subscriber;  
    private final InputStream inputStream;  
    private ExecutorService executor = Executors.newFixedThreadPool(1);  
    private AtomicLong demand = new AtomicLong(0);  
  
    SubscriptionImpl(Subscriber<? super AudioStream> s, InputStream inputStream)  
{  
      this.subscriber = s;  
      this.inputStream = inputStream;  
    }  
  
    @Override  
    public void request(long n) {  
      if (n <= 0) {  
        subscriber.onError(new IllegalArgumentException("Demand must be  
positive"));  
      }  
  
      demand.getAndAdd(n);  
  
      executor.submit(() -> {  
        try {  
          do {  
            ByteBuffer audioBuffer = getNextEvent();  
            if (audioBuffer.remaining() > 0) {  
              AudioEvent audioEvent =  
audioEventFromBuffer(audioBuffer);  
              subscriber.onNext(audioEvent);  
            } else {  
              subscriber.onComplete();  
              break;  
            }  
          } while (demand.decrementAndGet() > 0);  
        } catch (Exception e) {  
          subscriber.onError(e);  
        }  
      });  
    }  
  }  
}
```

```
@Override
public void cancel() {
    executor.shutdown();
}

private ByteBuffer getNextEvent() {
    ByteBuffer audioBuffer = null;
    byte[] audioBytes = new byte[CHUNK_SIZE_IN_BYTES];

    int len = 0;
    try {
        len = inputStream.read(audioBytes);

        if (len <= 0) {
            audioBuffer = ByteBuffer.allocate(0);
        } else {
            audioBuffer = ByteBuffer.wrap(audioBytes, 0, len);
        }
    } catch (IOException e) {
        throw new UncheckedIOException(e);
    }

    return audioBuffer;
}

private AudioEvent audioEventFromBuffer(ByteBuffer bb) {
    return AudioEvent.builder()
        .audioChunk(SdkBytes.fromByteBuffer(bb))
        .build();
}
}
```

- Per i dettagli sull'API, [StartTranscriptionJob](#) consulta AWS SDK for Java 2.x API Reference.

## Scenari

Trascrivere l'audio e ottenere i dati di processo

L'esempio di codice seguente mostra come:

- Avviare un processo di trascrizione con Amazon Transcribe.

- Attendi il completamento del processo .
- Ottenere l'URI in cui è archiviata la trascrizione.

Per ulteriori informazioni, consulta [Nozioni di base su Amazon Transcribe](#).

SDK per Java 2.x

#### Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Trascrive un file PCM.

```
/**
 * To run this AWS code example, ensure that you have set up your development
 * environment, including your AWS credentials.
 *
 * For information, see this documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class TranscribeStreamingDemoFile {
    private static final Region REGION = Region.US_EAST_1;
    private static TranscribeStreamingAsyncClient client;

    public static void main(String args[]) throws ExecutionException,
    InterruptedException {

        final String USAGE = "\n" +
            "Usage:\n" +
            "  <file> \n\n" +
            "Where:\n" +
            "  file - the location of a PCM file to transcribe. In this
example, ensure the PCM file is 16 hertz (Hz). \n";

        if (args.length != 1) {
            System.out.println(USAGE);
            System.exit(1);
        }
    }
}
```

```
    }

    String file = args[0];
    client = TranscribeStreamingAsyncClient.builder()
        .region(REGION)
        .build();

    CompletableFuture<Void> result =
client.startStreamTranscription(getRequest(16_000),
    new AudioStreamPublisher(getStreamFromFile(file)),
    getResponseHandler());

    result.get();
    client.close();
}

private static InputStream getStreamFromFile(String file) {
    try {
        File inputFile = new File(file);
        InputStream audioStream = new FileInputStream(inputFile);
        return audioStream;

    } catch (FileNotFoundException e) {
        throw new RuntimeException(e);
    }
}

private static StartStreamTranscriptionRequest getRequest(Integer
mediaSampleRateHertz) {
    return StartStreamTranscriptionRequest.builder()
        .languageCode(LanguageCode.EN_US)
        .mediaEncoding(MediaEncoding.PCM)
        .mediaSampleRateHertz(mediaSampleRateHertz)
        .build();
}

private static StartStreamTranscriptionResponseHandler getResponseHandler() {
    return StartStreamTranscriptionResponseHandler.builder()
        .onResponse(r -> {
            System.out.println("Received Initial response");
        })
        .onError(e -> {
            System.out.println(e.getMessage());
            StringWriter sw = new StringWriter();

```

```

        e.printStackTrace(new PrintWriter(sw));
        System.out.println("Error Occurred: " + sw.toString());
    })
    .onComplete(() -> {
        System.out.println("=== All records stream successfully ===");
    })
    .subscriber(event -> {
        List<Result> results = ((TranscriptEvent)
event).transcript().results();
        if (results.size() > 0) {
            if (!
results.get(0).alternatives().get(0).transcript().isEmpty()) {

System.out.println(results.get(0).alternatives().get(0).transcript());
            }
        }
    })
    .build();
}

private static class AudioStreamPublisher implements Publisher<AudioStream> {
    private final InputStream inputStream;
    private static Subscription currentSubscription;

    private AudioStreamPublisher(InputStream inputStream) {
        this.inputStream = inputStream;
    }

    @Override
    public void subscribe(Subscriber<? super AudioStream> s) {

        if (this.currentSubscription == null) {
            this.currentSubscription = new SubscriptionImpl(s, inputStream);
        } else {
            this.currentSubscription.cancel();
            this.currentSubscription = new SubscriptionImpl(s, inputStream);
        }
        s.onSubscribe(currentSubscription);
    }
}

public static class SubscriptionImpl implements Subscription {
    private static final int CHUNK_SIZE_IN_BYTES = 1024 * 1;
    private final Subscriber<? super AudioStream> subscriber;

```

```
private final InputStream inputStream;
private ExecutorService executor = Executors.newFixedThreadPool(1);
private AtomicLong demand = new AtomicLong(0);

SubscriptionImpl(Subscriber<? super AudioStream> s, InputStream inputStream)
{
    this.subscriber = s;
    this.inputStream = inputStream;
}

@Override
public void request(long n) {
    if (n <= 0) {
        subscriber.onError(new IllegalArgumentException("Demand must be
positive"));
    }

    demand.getAndAdd(n);

    executor.submit(() -> {
        try {
            do {
                ByteBuffer audioBuffer = getNextEvent();
                if (audioBuffer.remaining() > 0) {
                    AudioEvent audioEvent =
audioEventFromBuffer(audioBuffer);
                    subscriber.onNext(audioEvent);
                } else {
                    subscriber.onComplete();
                    break;
                }
            } while (demand.decrementAndGet() > 0);
        } catch (Exception e) {
            subscriber.onError(e);
        }
    });
}

@Override
public void cancel() {
    executor.shutdown();
}

private ByteBuffer getNextEvent() {
```



```

        ByteBuffer audioBuffer = null;
        byte[] audioBytes = new byte[CHUNK_SIZE_IN_BYTES];

        int len = 0;
        try {
            len = inputStream.read(audioBytes);

            if (len <= 0) {
                audioBuffer = ByteBuffer.allocate(0);
            } else {
                audioBuffer = ByteBuffer.wrap(audioBytes, 0, len);
            }
        } catch (IOException e) {
            throw new UncheckedIOException(e);
        }

        return audioBuffer;
    }

    private AudioEvent audioEventFromBuffer(ByteBuffer bb) {
        return AudioEvent.builder()
            .audioChunk(SdkBytes.fromByteBuffer(bb))
            .build();
    }
}

```

Trascrive l'audio in streaming dal microfono del computer.

```

public class TranscribeStreamingDemoApp {
    private static final Region REGION = Region.US_EAST_1;
    private static TranscribeStreamingAsyncClient client;

    public static void main(String args[])
        throws URISyntaxException, ExecutionException, InterruptedException,
        LineUnavailableException {

        client = TranscribeStreamingAsyncClient.builder()
            .credentialsProvider(getCredentials())
            .region(REGION)
            .build();
    }
}

```

```
        CompletableFuture<Void> result =
client.startStreamTranscription(getRequest(16_000),
    new AudioStreamPublisher(getStreamFromMic()),
    getResponseHandler());

    result.get();
    client.close();
}

private static InputStream getStreamFromMic() throws LineUnavailableException {

    // Signed PCM AudioFormat with 16kHz, 16 bit sample size, mono
    int sampleRate = 16000;
    AudioFormat format = new AudioFormat(sampleRate, 16, 1, true, false);
    DataLine.Info info = new DataLine.Info(TargetDataLine.class, format);

    if (!AudioSystem.isLineSupported(info)) {
        System.out.println("Line not supported");
        System.exit(0);
    }

    TargetDataLine line = (TargetDataLine) AudioSystem.getLine(info);
    line.open(format);
    line.start();

    InputStream audioStream = new AudioInputStream(line);
    return audioStream;
}

private static AwsCredentialsProvider getCredentials() {
    return DefaultCredentialsProvider.create();
}

private static StartStreamTranscriptionRequest getRequest(Integer
mediaSampleRateHertz) {
    return StartStreamTranscriptionRequest.builder()
        .languageCode(LanguageCode.EN_US.toString())
        .mediaEncoding(MediaEncoding.PCM)
        .mediaSampleRateHertz(mediaSampleRateHertz)
        .build();
}

private static StartStreamTranscriptionResponseHandler getResponseHandler() {
    return StartStreamTranscriptionResponseHandler.builder()

```

```

        .onResponse(r -> {
            System.out.println("Received Initial response");
        })
        .onError(e -> {
            System.out.println(e.getMessage());
            StringWriter sw = new StringWriter();
            e.printStackTrace(new PrintWriter(sw));
            System.out.println("Error Occurred: " + sw.toString());
        })
        .onComplete(() -> {
            System.out.println("=== All records stream successfully ===");
        })
        .subscriber(event -> {
            List<Result> results = ((TranscriptEvent)
event).transcript().results();
            if (results.size() > 0) {
                if (!
results.get(0).alternatives().get(0).transcript().isEmpty()) {

System.out.println(results.get(0).alternatives().get(0).transcript());
                }
            }
        })
        .build();
    }

    private InputStream getStreamFromFile(String audioFileName) {
        try {
            File inputFile = new
File(getClass().getClassLoader().getResource(audioFileName).getFile());
            InputStream audioStream = new FileInputStream(inputFile);
            return audioStream;
        } catch (FileNotFoundException e) {
            throw new RuntimeException(e);
        }
    }

    private static class AudioStreamPublisher implements Publisher<AudioStream> {
        private final InputStream inputStream;
        private static Subscription currentSubscription;

        private AudioStreamPublisher(InputStream inputStream) {
            this.inputStream = inputStream;
        }
    }

```

```

@Override
public void subscribe(Subscriber<? super AudioStream> s) {

    if (this.currentSubscription == null) {
        this.currentSubscription = new SubscriptionImpl(s, inputStream);
    } else {
        this.currentSubscription.cancel();
        this.currentSubscription = new SubscriptionImpl(s, inputStream);
    }
    s.onSubscribe(currentSubscription);
}

}

public static class SubscriptionImpl implements Subscription {
    private static final int CHUNK_SIZE_IN_BYTES = 1024 * 1;
    private final Subscriber<? super AudioStream> subscriber;
    private final InputStream inputStream;
    private ExecutorService executor = Executors.newFixedThreadPool(1);
    private AtomicLong demand = new AtomicLong(0);

    SubscriptionImpl(Subscriber<? super AudioStream> s, InputStream inputStream)
{
        this.subscriber = s;
        this.inputStream = inputStream;
    }

    @Override
    public void request(long n) {
        if (n <= 0) {
            subscriber.onError(new IllegalArgumentException("Demand must be
positive"));
        }

        demand.getAndAdd(n);

        executor.submit(() -> {
            try {
                do {
                    ByteBuffer audioBuffer = getNextEvent();
                    if (audioBuffer.remaining() > 0) {
                        AudioEvent audioEvent =
audioEventFromBuffer(audioBuffer);
                        subscriber.onNext(audioEvent);
                    }
                } while (demand.get() > 0);
            } catch (Exception e) {
                subscriber.onError(e);
            }
        });
    }
}

```

```
        } else {
            subscriber.onComplete();
            break;
        }
    } while (demand.decrementAndGet() > 0);
} catch (Exception e) {
    subscriber.onError(e);
}
});
}

@Override
public void cancel() {
    executor.shutdown();
}

private ByteBuffer getNextEvent() {
    ByteBuffer audioBuffer = null;
    byte[] audioBytes = new byte[CHUNK_SIZE_IN_BYTES];

    int len = 0;
    try {
        len = inputStream.read(audioBytes);

        if (len <= 0) {
            audioBuffer = ByteBuffer.allocate(0);
        } else {
            audioBuffer = ByteBuffer.wrap(audioBytes, 0, len);
        }
    } catch (IOException e) {
        throw new UncheckedIOException(e);
    }

    return audioBuffer;
}

private AudioEvent audioEventFromBuffer(ByteBuffer bb) {
    return AudioEvent.builder()
        .audioChunk(SdkBytes.fromByteBuffer(bb))
        .build();
}
}
}
```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella Documentazione di riferimento delle API AWS SDK for Java 2.x .
  - [GetTranscriptionJob](#)
  - [StartTranscriptionJob](#)

## Esempi cross-service che utilizzano SDK for Java 2.x

Le seguenti applicazioni di esempio utilizzano il AWS SDK for Java 2.x per funzionare su più applicazioni Servizi AWS.

Gli esempi trasversali mirano a un livello avanzato di esperienza per aiutarti a iniziare a creare applicazioni.

### Esempi

- [Costruisci un'applicazione per inviare dati a una tabella DynamoDB](#)
- [Crea un chatbot Amazon Lex per coinvolgere i visitatori del tuo sito web](#)
- [Costruzione di un'applicazione per la pubblicazione e la sottoscrizione che traduce i messaggi](#)
- [Crea un'applicazione Web che invia e recupera messaggi utilizzando Amazon SQS](#)
- [Creazione di un'applicazione di gestione delle risorse fotografiche che consente agli utenti di gestire le foto utilizzando etichette](#)
- [Creazione di un'applicazione Web per tracciare i dati DynamoDB](#)
- [Come creare un tracker di articoli Amazon Redshift](#)
- [Creazione di un tracciatore di elementi di lavoro di Aurora Serverless](#)
- [Crea un'applicazione che analizza il feedback dei clienti e sintetizza l'audio](#)
- [Rileva i DPI nelle immagini con Amazon Rekognition utilizzando un SDK AWS](#)
- [Rileva oggetti nelle immagini con Amazon Rekognition utilizzando un SDK AWS](#)
- [Rileva persone e oggetti in un video con Amazon Rekognition utilizzando un SDK AWS](#)
- [Monitora le prestazioni di Amazon DynamoDB utilizzando un SDK AWS](#)
- [Pubblica messaggi Amazon SNS nelle code Amazon SQS utilizzando un SDK AWS](#)
- [Utilizzo di un'API Gateway per richiamare una funzione Lambda](#)
- [Utilizzo di Step Functions per richiamare le funzioni Lambda](#)

- [Utilizzo degli eventi pianificati per richiamare una funzione Lambda](#)

## Costruisci un'applicazione per inviare dati a una tabella DynamoDB

### SDK per Java 2.x

Mostra come creare un'applicazione Web dinamica che invia dati utilizzando l'API Java di Amazon DynamoDB e invia un messaggio di testo utilizzando l'API Java di Amazon Simple Notification Service.

Per il codice sorgente completo e le istruzioni su come configurarlo ed eseguirlo, guarda l'esempio completo su [GitHub](#).

Servizi utilizzati in questo esempio

- DynamoDB
- Amazon SNS

## Crea un chatbot Amazon Lex per coinvolgere i visitatori del tuo sito web

### SDK per Java 2.x

Mostra come utilizzare l'API Amazon Lex per creare un Chatbot all'interno di un'applicazione Web per coinvolgere i visitatori del sito Web.

Per il codice sorgente completo e le istruzioni su come configurarlo ed eseguirlo, consulta l'esempio completo su [GitHub](#).

Servizi utilizzati in questo esempio

- Amazon Comprehend
- Amazon Lex
- Amazon Translate

## Costruzione di un'applicazione per la pubblicazione e la sottoscrizione che traduce i messaggi

### SDK per Java 2.x

Mostra come utilizzare l'API Java di Amazon Simple Notification Service per creare un'applicazione Web con funzionalità di sottoscrizione e pubblicazione. Inoltre, questa applicazione di esempio traduce anche i messaggi.

Per il codice sorgente completo e le istruzioni su come configurarlo ed eseguirlo, guarda l'esempio completo su [GitHub](#).

Per il codice sorgente completo e le istruzioni su come configurare ed eseguire l'esempio che utilizza l'API Java Async, vedi l'esempio completo su [GitHub](#).

Servizi utilizzati in questo esempio

- Amazon SNS
- Amazon Translate

## Crea un'applicazione Web che invia e recupera messaggi utilizzando Amazon SQS

### SDK per Java 2.x

Mostra come utilizzare l'API Amazon SQS per sviluppare un'API REST Spring che invia e recupera messaggi.

Per il codice sorgente completo e le istruzioni su come configurarlo ed eseguirlo, consulta l'esempio completo su [GitHub](#).

Servizi utilizzati in questo esempio

- Amazon Comprehend
- Amazon SQS



## Creazione di un'applicazione di gestione delle risorse fotografiche che consente agli utenti di gestire le foto utilizzando etichette

### SDK per Java 2.x

Mostra come sviluppare un'applicazione per la gestione delle risorse fotografiche che rileva le etichette nelle immagini utilizzando Amazon Rekognition e le archivia per recuperarle in seguito.

Per il codice sorgente completo e le istruzioni su come configurarlo ed eseguirlo, guarda l'esempio completo su [GitHub](#).

Per approfondire l'origine di questo esempio, consulta il post su [AWS Community](#).

Servizi utilizzati in questo esempio

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

## Creazione di un'applicazione Web per tracciare i dati DynamoDB

### SDK per Java 2.x

Mostra come utilizzare l'API Amazon DynamoDB per creare un'applicazione Web dinamica che traccia i dati di lavoro DynamoDB.

Per il codice sorgente completo e le istruzioni su come configurarlo ed eseguirlo, guarda l'esempio completo su [GitHub](#).

Servizi utilizzati in questo esempio

- DynamoDB
- Amazon SES

## Come creare un tracker di articoli Amazon Redshift

### SDK per Java 2.x

Mostra come creare un'applicazione Web che traccia e segnala gli elementi di lavoro archiviati in un database Amazon Redshift.

Per il codice sorgente completo e le istruzioni su come configurare un'API Spring REST che interroga i dati di Amazon Redshift e per l'utilizzo da parte di un'applicazione React, consulta l'esempio completo su. [GitHub](#)

Servizi utilizzati in questo esempio

- Amazon Redshift
- Amazon SES

## Creazione di un tracciatore di elementi di lavoro di Aurora Serverless

### SDK per Java 2.x

Mostra come creare un'applicazione Web che traccia e segnala gli elementi di lavoro archiviati in un database Amazon RDS.

Per il codice sorgente completo e le istruzioni su come configurare un'API Spring REST che interroga i dati Serverless di Amazon Aurora e per l'utilizzo da parte di un'applicazione React, consulta l'esempio completo su. [GitHub](#)

Per il codice sorgente completo e le istruzioni su come configurare ed eseguire un esempio che utilizza l'API JDBC, consulta l'esempio completo su. [GitHub](#)

Servizi utilizzati in questo esempio

- Aurora
- Amazon RDS
- Servizi di dati di Amazon RDS
- Amazon SES

# Crea un'applicazione che analizza il feedback dei clienti e sintetizza l'audio

## SDK per Java 2.x

Questa applicazione di esempio analizza e archivia le schede di feedback dei clienti. In particolare, soddisfa l'esigenza di un hotel fittizio a New York City. L'hotel riceve feedback dagli ospiti in varie lingue sotto forma di schede di commento fisiche. Tale feedback viene caricato nell'app tramite un client Web. Dopo aver caricato l'immagine di una scheda di commento, vengono eseguiti i seguenti passaggi:

- Il testo viene estratto dall'immagine utilizzando Amazon Textract.
- Amazon Comprehend determina il sentiment del testo estratto e la sua lingua.
- Il testo estratto viene tradotto in inglese utilizzando Amazon Translate.
- Amazon Polly sintetizza un file audio dal testo estratto.

L'app completa può essere implementata con AWS CDK. Per il codice sorgente e le istruzioni di distribuzione, consulta il progetto in [GitHub](#)

### Servizi utilizzati in questo esempio

- Amazon Comprehend
- Lambda
- Amazon Polly
- Amazon Textract
- Amazon Translate

# Rileva i DPI nelle immagini con Amazon Rekognition utilizzando un SDK AWS

## SDK per Java 2.x

Mostra come creare una AWS Lambda funzione che rileva le immagini con dispositivi di protezione individuale.

Per il codice sorgente completo e le istruzioni su come configurarlo ed eseguirlo, guarda l'esempio completo su [GitHub](#).

Servizi utilizzati in questo esempio

- DynamoDB
- Amazon Rekognition
- Amazon S3
- Amazon SES

## Rileva oggetti nelle immagini con Amazon Rekognition utilizzando un SDK AWS

SDK per Java 2.x

Mostra come utilizzare l'API Java di Amazon Rekognition per creare un'applicazione che utilizza Amazon Rekognition per identificare gli oggetti in base a una categoria nelle immagini situate in un bucket Amazon Simple Storage Service (Amazon S3). L'applicazione invia all'amministratore una notifica e-mail sui risultati tramite Amazon Simple Email Service (Amazon SES).

Per il codice sorgente completo e le istruzioni su come configurarlo ed eseguirlo, consulta l'esempio completo su. [GitHub](#)

Servizi utilizzati in questo esempio

- Amazon Rekognition
- Amazon S3
- Amazon SES

## Rileva persone e oggetti in un video con Amazon Rekognition utilizzando un SDK AWS

SDK per Java 2.x

Mostra come utilizzare l'API Java di Amazon Rekognition per creare un'applicazione che rileva volti e oggetti nei video situati in un bucket Amazon Simple Storage Service (Amazon S3). L'applicazione invia all'amministratore una notifica e-mail sui risultati tramite Amazon Simple Email Service (Amazon SES).

Per il codice sorgente completo e le istruzioni su come configurarlo ed eseguirlo, guarda l'esempio completo su. [GitHub](#)

### Servizi utilizzati in questo esempio

- Amazon Rekognition
- Amazon S3
- Amazon SES

## Monitora le prestazioni di Amazon DynamoDB utilizzando un SDK AWS

### SDK per Java 2.x

Questo esempio mostra come configurare un'applicazione Java per monitorare le prestazioni di DynamoDB. L'applicazione invia i dati metrici a CloudWatch cui è possibile monitorare le prestazioni.

Per il codice sorgente completo e le istruzioni su come configurarlo ed eseguirlo, guarda l'esempio completo su [GitHub](#).

### Servizi utilizzati in questo esempio

- CloudWatch
- DynamoDB

## Pubblica messaggi Amazon SNS nelle code Amazon SQS utilizzando un SDK AWS

### SDK per Java 2.x

Illustrazione della messaggistica con argomenti e code utilizzando Amazon Simple Notification Service (Amazon SNS) e Amazon Simple Queue Service (Amazon SQS).

Per il codice sorgente completo e le istruzioni che illustrano la messaggistica con argomenti e code in Amazon SNS e Amazon SQS, consulta l'esempio completo su [GitHub](#)

### Servizi utilizzati in questo esempio

- Amazon SNS
- Amazon SQS

## Utilizzo di un'API Gateway per richiamare una funzione Lambda

### SDK per Java 2.x

Mostra come creare una AWS Lambda funzione utilizzando l'API runtime Lambda Java. Questo esempio richiama diversi AWS servizi per eseguire un caso d'uso specifico. In questo esempio viene illustrato come creare una funzione Lambda richiamata da Gateway Amazon API che analizza una tabella Amazon DynamoDB per le ricorrenze di lavoro e utilizza Amazon Simple Notification Service (Amazon SNS) per inviare un messaggio di testo ai dipendenti che si congratula con loro alla data dell'anniversario di un anno.

Per il codice sorgente completo e le istruzioni su come configurarlo ed eseguirlo, guarda l'esempio completo su [GitHub](#)

Servizi utilizzati in questo esempio

- API Gateway
- DynamoDB
- Lambda
- Amazon SNS

## Utilizzo di Step Functions per richiamare le funzioni Lambda

### SDK per Java 2.x

Mostra come creare un flusso di lavoro AWS serverless utilizzando AWS Step Functions and. AWS SDK for Java 2.x Ogni fase del flusso di lavoro viene implementata utilizzando una AWS Lambda funzione.

Per il codice sorgente completo e le istruzioni su come configurarlo ed eseguirlo, vedi l'esempio completo su [GitHub](#).

Servizi utilizzati in questo esempio

- DynamoDB
- Lambda
- Amazon SES
- Step Functions

# Utilizzo degli eventi pianificati per richiamare una funzione Lambda

## SDK per Java 2.x

Mostra come creare un evento EventBridge pianificato da Amazon che richiami una AWS Lambda funzione. Configura EventBridge per utilizzare un'espressione cron per pianificare quando viene richiamata la funzione Lambda. In questo esempio, viene creata una funzione Lambda utilizzando l'API di runtime Lambda Java. Questo esempio richiama diversi AWS servizi per eseguire un caso d'uso specifico. Questo esempio dimostra come creare un'app che invia un messaggio di testo via mobile ai tuoi dipendenti che si congratula con loro alla data dell'anniversario di un anno.

Per il codice sorgente completo e le istruzioni su come configurarlo ed eseguirlo, guarda l'esempio completo su. [GitHub](#)

Servizi utilizzati in questo esempio

- DynamoDB
- EventBridge
- Lambda
- Amazon SNS

# Sicurezza per AWS SDK for Java

La sicurezza cloud di Amazon Web Services (AWS) è la priorità più alta. In quanto cliente AWS, è possibile trarre vantaggio da un'architettura di data center e di rete progettata per soddisfare i requisiti delle organizzazioni più esigenti a livello di sicurezza. La sicurezza è una responsabilità condivisa tra te e te. AWS Il [modello di responsabilità condivisa](#) descrive questo come sicurezza del cloud e sicurezza nel cloud.

Security of the Cloud: AWS è responsabile della protezione dell'infrastruttura che gestisce tutti i servizi offerti nel AWS Cloud e della fornitura di servizi che è possibile utilizzare in modo sicuro. La nostra responsabilità in AWS materia di sicurezza è la massima priorità e l'efficacia della nostra sicurezza viene regolarmente testata e verificata da revisori di terze parti nell'ambito dei Programmi di [AWS conformità](#).

Sicurezza nel cloud: la responsabilità dell'utente è determinata dal AWS servizio utilizzato e da altri fattori, tra cui la sensibilità dei dati, i requisiti dell'organizzazione e le leggi e i regolamenti applicabili.

Questo AWS prodotto o servizio segue il [modello di responsabilità condivisa](#) attraverso i servizi specifici di Amazon Web Services (AWS) che supporta. Per informazioni sulla sicurezza dei AWS servizi, consulta la [pagina della documentazione sulla sicurezza del AWS servizio](#) e [AWS i servizi che rientrano nell'ambito delle iniziative di AWS conformità previste dal programma di conformità](#).

## Argomenti

- [Protezione dei dati in AWS SDK for Java 2.x](#)
- [Utilizzo di TLS nell'SDK for Java](#)
- [Identity and Access Management](#)
- [Convalida della conformità per questo AWS prodotto o servizio](#)
- [Resilienza per questo AWS prodotto o servizio](#)
- [Sicurezza dell'infrastruttura per questo AWS prodotto o servizio](#)

## Protezione dei dati in AWS SDK for Java 2.x

Il modello di [responsabilità AWS condivisa modello](#) di di si applica alla protezione dei dati in AWS SDK for Java. Come descritto in questo modello, AWS è responsabile della protezione dell'infrastruttura globale che gestisce tutti i Cloud AWS. L'utente è responsabile del controllo dei



contenuti ospitati su questa infrastruttura. L'utente è inoltre responsabile della configurazione della protezione e delle attività di gestione per i Servizi AWS utilizzati. Per ulteriori informazioni sulla privacy dei dati, vedi le [Domande frequenti sulla privacy dei dati](#). Per informazioni sulla protezione dei dati in Europa, consulta il post del blog relativo al [Modello di responsabilità condivisa AWS e GDPR](#) nel Blog sulla sicurezza AWS .

Ai fini della protezione dei dati, consigliamo di proteggere Account AWS le credenziali e configurare i singoli utenti con AWS IAM Identity Center or AWS Identity and Access Management (IAM). In tal modo, a ogni utente verranno assegnate solo le autorizzazioni necessarie per svolgere i suoi compiti. Ti suggeriamo, inoltre, di proteggere i dati nei seguenti modi:

- Utilizza l'autenticazione a più fattori (MFA) con ogni account.
- Usa SSL/TLS per comunicare con le risorse. AWS È richiesto TLS 1.2 ed è consigliato TLS 1.3.
- Configura l'API e la registrazione delle attività degli utenti con. AWS CloudTrail
- Utilizza soluzioni di AWS crittografia, insieme a tutti i controlli di sicurezza predefiniti all'interno Servizi AWS.
- Utilizza i servizi di sicurezza gestiti avanzati, come Amazon Macie, che aiutano a individuare e proteggere i dati sensibili archiviati in Amazon S3.
- Se hai bisogno di moduli crittografici convalidati FIPS 140-2 per l'accesso AWS tramite un'interfaccia a riga di comando o un'API, utilizza un endpoint FIPS. Per ulteriori informazioni sugli endpoint FIPS disponibili, consulta il [Federal Information Processing Standard \(FIPS\) 140-2](#).

Ti consigliamo vivamente di non inserire mai informazioni riservate o sensibili, ad esempio gli indirizzi e-mail dei clienti, nei tag o nei campi di testo in formato libero, ad esempio nel campo Nome. Ciò include quando lavori con SDK for Java o Servizi AWS altro utilizzando la console, l'API AWS CLI AWS o gli SDK. I dati inseriti nei tag o nei campi di testo in formato libero utilizzati per i nomi possono essere utilizzati per i la fatturazione o i log di diagnostica. Quando fornisci un URL a un server esterno, ti suggeriamo vivamente di non includere informazioni sulle credenziali nell'URL per convalidare la tua richiesta al server.

## Utilizzo di TLS nell'SDK for Java

AWS SDK for Java Utilizza le funzionalità TLS della piattaforma Java sottostante. [In questo argomento, mostriamo esempi di utilizzo dell'implementazione OpenJDK utilizzata da Amazon Corretto 17.](#)

Per utilizzarlo Servizi AWS, il JDK sottostante deve supportare una versione minima di TLS 1.2, ma TLS 1.3 è consigliato.

Gli utenti devono consultare la documentazione della piattaforma Java che utilizzano con l'SDK per scoprire quali versioni TLS sono abilitate di default e come abilitare e disabilitare versioni TLS specifiche.

## Come controllare le informazioni sulla versione TLS

[Utilizzando OpenJDK, il codice seguente mostra l'uso di SSLContext per stampare quali versioni TLS/SSL sono supportate.](#)

```
System.out.println(Arrays.toString(SSLContext.getDefault().getSupportedSSLParameters().getProtocols()));
```

Ad esempio, Amazon Corretto 17 (OpenJDK) produce il seguente output.

```
[TLSv1.3, TLSv1.2, TLSv1.1, TLSv1, SSLv3, SSLv2Hello]
```

Per vedere l'handshake SSL in azione e quale versione di TLS viene utilizzata, puoi utilizzare la proprietà di sistema `javax.net.debug`.

Ad esempio, esegui un'applicazione Java che utilizza TLS.

```
java app.jar -Djavax.net.debug=ssl:handshake
```

L'applicazione registra l'handshake SSL in modo simile al seguente.

```
...
javax.net.ssl|DEBUG|10|main|2022-12-23 13:53:12.221 EST|ClientHello.java:641|Produced
  ClientHello handshake message (
  "ClientHello": {
    "client version"      : "TLSv1.2",
    ...
  }
  javax.net.ssl|DEBUG|10|main|2022-12-23 13:53:12.295 EST|ServerHello.java:888|Consuming
  ServerHello handshake message (
  "ServerHello": {
    "server version"     : "TLSv1.2",
    ...
  }
  ...
```

## Applica una versione TLS minima

L'SDK for Java preferisce sempre l'ultima versione TLS supportata dalla piattaforma e dal servizio. Se desideri applicare una versione TLS minima specifica, consulta la documentazione della tua piattaforma Java.

Per le JVM basate su OpenJDK, è possibile utilizzare la proprietà `system.`

```
jdk.tls.client.protocols
```

Ad esempio, se desideri che i client del servizio SDK nell'applicazione utilizzino TLS 1.2, anche se TLS 1.3 è disponibile, fornisci la seguente proprietà di sistema.

```
java app.jar -Djdk.tls.client.protocols=TLSv1.2
```

## AWS Aggiornamento degli endpoint API a TLS 1.2

Consulta questo [post di blog](#) per informazioni sugli endpoint AWS API che passano a TLS 1.2 per la versione minima.

## Identity and Access Management

AWS Identity and Access Management (IAM) è un software Servizio AWS che aiuta un amministratore a controllare in modo sicuro l'accesso alle AWS risorse. Gli amministratori IAM controllano chi può essere autenticato (effettuato l'accesso) e autorizzato (disporre delle autorizzazioni) a utilizzare le risorse. AWS IAM è uno Servizio AWS strumento che puoi utilizzare senza costi aggiuntivi.

### Argomenti

- [Destinatari](#)
- [Autenticazione con identità](#)
- [Gestione dell'accesso con policy](#)
- [Come Servizi AWS lavorare con IAM](#)
- [Risoluzione dei problemi di AWS identità e accesso](#)

## Destinatari

Il modo in cui usi AWS Identity and Access Management (IAM) varia a seconda del lavoro che AWS svolgi.

**Utente del servizio:** se lo utilizzi Servizi AWS per svolgere il tuo lavoro, l'amministratore ti fornisce le credenziali e le autorizzazioni necessarie. Man mano che utilizzi più AWS funzionalità per svolgere il tuo lavoro, potresti aver bisogno di autorizzazioni aggiuntive. La comprensione della gestione dell'accesso ti consente di richiedere le autorizzazioni corrette all'amministratore. Se non riesci ad accedere a una funzionalità di AWS, consulta [Risoluzione dei problemi di AWS identità e accesso](#) o consulta la guida per l'utente della funzionalità Servizio AWS che stai utilizzando.

**Amministratore del servizio:** se sei responsabile delle AWS risorse della tua azienda, probabilmente hai pieno accesso a AWS. È tuo compito determinare a quali AWS funzionalità e risorse devono accedere gli utenti del servizio. Devi inviare le richieste all'amministratore IAM per cambiare le autorizzazioni degli utenti del servizio. Esamina le informazioni contenute in questa pagina per comprendere i concetti di base relativi a IAM. Per saperne di più su come la tua azienda può utilizzare IAM con AWS, consulta la guida per l'utente del Servizio AWS software che stai utilizzando.

**Amministratore IAM:** un amministratore IAM potrebbe essere interessato a ottenere dei dettagli su come scrivere policy per gestire l'accesso a AWS. Per visualizzare esempi di policy AWS basate sull'identità che puoi utilizzare in IAM, consulta la guida per l'utente di quella Servizio AWS che stai utilizzando.

## Autenticazione con identità

L'autenticazione è il modo in cui accedi AWS utilizzando le tue credenziali di identità. Devi essere autenticato (aver effettuato l' Utente root dell'account AWS accesso AWS) come utente IAM o assumendo un ruolo IAM.

Puoi accedere AWS come identità federata utilizzando le credenziali fornite tramite una fonte di identità. AWS IAM Identity Center Gli utenti (IAM Identity Center), l'autenticazione Single Sign-On della tua azienda e le tue credenziali di Google o Facebook sono esempi di identità federate. Se accedi come identità federata, l'amministratore ha configurato in precedenza la federazione delle identità utilizzando i ruoli IAM. Quando accedi AWS utilizzando la federazione, assumi indirettamente un ruolo.

A seconda del tipo di utente, puoi accedere al AWS Management Console o al portale di AWS accesso. Per ulteriori informazioni sull'accesso a AWS, vedi [Come accedere al tuo Account AWS nella](#) Guida per l'Accedi ad AWS utente.

Se accedi a AWS livello di codice, AWS fornisce un kit di sviluppo software (SDK) e un'interfaccia a riga di comando (CLI) per firmare crittograficamente le tue richieste utilizzando le tue credenziali. Se non utilizzi AWS strumenti, devi firmare tu stesso le richieste. Per ulteriori informazioni sull'utilizzo del metodo consigliato per firmare autonomamente le richieste, consulta [Signing AWS API request](#) nella IAM User Guide.

A prescindere dal metodo di autenticazione utilizzato, potrebbe essere necessario specificare ulteriori informazioni sulla sicurezza. Ad esempio, ti AWS consiglia di utilizzare l'autenticazione a più fattori (MFA) per aumentare la sicurezza del tuo account. Per ulteriori informazioni, consulta [Autenticazione a più fattori](#) nella Guida per l'utente di AWS IAM Identity Center e [Utilizzo dell'autenticazione a più fattori \(MFA\) in AWS](#) nella Guida per l'utente di IAM.

## Account AWS utente root

Quando si crea un account Account AWS, si inizia con un'identità di accesso che ha accesso completo a tutte Servizi AWS le risorse dell'account. Questa identità è denominata utente Account AWS root ed è accessibile effettuando l'accesso con l'indirizzo e-mail e la password utilizzati per creare l'account. Si consiglia vivamente di non utilizzare l'utente root per le attività quotidiane. Conservare le credenziali dell'utente root e utilizzarle per eseguire le operazioni che solo l'utente root può eseguire. Per un elenco completo delle attività che richiedono l'accesso come utente root, consulta la sezione [Attività che richiedono le credenziali dell'utente root](#) nella Guida per l'utente di IAM.

## Identità federata

Come procedura consigliata, richiedi agli utenti umani, compresi gli utenti che richiedono l'accesso come amministratore, di utilizzare la federazione con un provider di identità per accedere Servizi AWS utilizzando credenziali temporanee.

Un'identità federata è un utente dell'elenco utenti aziendale, di un provider di identità Web AWS Directory Service, della directory Identity Center o di qualsiasi utente che accede utilizzando le Servizi AWS credenziali fornite tramite un'origine di identità. Quando le identità federate accedono Account AWS, assumono ruoli e i ruoli forniscono credenziali temporanee.

Per la gestione centralizzata degli accessi, consigliamo di utilizzare AWS IAM Identity Center. Puoi creare utenti e gruppi in IAM Identity Center oppure puoi connetterti e sincronizzarti con un set di utenti e gruppi nella tua fonte di identità per utilizzarli su tutte le tue applicazioni. Account AWS Per ulteriori informazioni sul Centro identità IAM, consulta [Cos'è Centro identità IAM?](#) nella Guida per l'utente di AWS IAM Identity Center .

## Utenti e gruppi IAM

Un [utente IAM](#) è un'identità interna Account AWS che dispone di autorizzazioni specifiche per una singola persona o applicazione. Ove possibile, consigliamo di fare affidamento a credenziali temporanee invece di creare utenti IAM con credenziali a lungo termine come le password e le chiavi di accesso. Tuttavia, per casi d'uso specifici che richiedono credenziali a lungo termine con utenti IAM, si consiglia di ruotare le chiavi di accesso. Per ulteriori informazioni, consulta la pagina [Rotazione periodica delle chiavi di accesso per casi d'uso che richiedono credenziali a lungo termine](#) nella Guida per l'utente di IAM.

Un [gruppo IAM](#) è un'identità che specifica un insieme di utenti IAM. Non è possibile eseguire l'accesso come gruppo. È possibile utilizzare gruppi per specificare le autorizzazioni per più utenti alla volta. I gruppi semplificano la gestione delle autorizzazioni per set di utenti di grandi dimensioni. Ad esempio, è possibile avere un gruppo denominato IAMAdmins e concedere a tale gruppo le autorizzazioni per amministrare le risorse IAM.

Gli utenti sono diversi dai ruoli. Un utente è associato in modo univoco a una persona o un'applicazione, mentre un ruolo è destinato a essere assunto da chiunque ne abbia bisogno. Gli utenti dispongono di credenziali a lungo termine permanenti, mentre i ruoli forniscono credenziali temporanee. Per ulteriori informazioni, consulta [Quando creare un utente IAM \(invece di un ruolo\)](#) nella Guida per l'utente di IAM.

## Ruoli IAM

Un [ruolo IAM](#) è un'identità interna all'utente Account AWS che dispone di autorizzazioni specifiche. È simile a un utente IAM, ma non è associato a una persona specifica. Puoi assumere temporaneamente un ruolo IAM in AWS Management Console [cambiando ruolo](#). Puoi assumere un ruolo chiamando un'operazione AWS CLI o AWS API o utilizzando un URL personalizzato. Per ulteriori informazioni sui metodi per l'utilizzo dei ruoli, consulta [Utilizzo di ruoli IAM](#) nella Guida per l'utente di IAM.

I ruoli IAM con credenziali temporanee sono utili nelle seguenti situazioni:

- **Accesso utente federato:** per assegnare le autorizzazioni a una identità federata, è possibile creare un ruolo e definire le autorizzazioni per il ruolo. Quando un'identità federata viene autenticata, l'identità viene associata al ruolo e ottiene le autorizzazioni da esso definite. Per ulteriori informazioni sulla federazione dei ruoli, consulta [Creazione di un ruolo per un provider di identità di terza parte](#) nella Guida per l'utente di IAM. Se utilizzi IAM Identity Center, configura un set di autorizzazioni. IAM Identity Center mette in correlazione il set di autorizzazioni con un

ruolo in IAM per controllare a cosa possono accedere le identità dopo l'autenticazione. Per ulteriori informazioni sui set di autorizzazioni, consulta [Set di autorizzazioni](#) nella Guida per l'utente di AWS IAM Identity Center .

- **Autorizzazioni utente IAM temporanee:** un utente IAM o un ruolo può assumere un ruolo IAM per ottenere temporaneamente autorizzazioni diverse per un'attività specifica.
- **Accesso multi-account:** è possibile utilizzare un ruolo IAM per permettere a un utente (un principale affidabile) con un account diverso di accedere alle risorse nell'account. I ruoli sono lo strumento principale per concedere l'accesso multi-account. Tuttavia, con alcuni Servizi AWS, è possibile allegare una policy direttamente a una risorsa (anziché utilizzare un ruolo come proxy). Per informazioni sulle differenze tra ruoli e policy basate su risorse per l'accesso multi-account, consulta [Differenza tra i ruoli IAM e le policy basate su risorse](#) nella Guida per l'utente di IAM.
- **Accesso a più servizi:** alcuni Servizi AWS utilizzano le funzionalità di altri Servizi AWS. Ad esempio, quando effettui una chiamata in un servizio, è comune che tale servizio esegua applicazioni in Amazon EC2 o archivi oggetti in Amazon S3. Un servizio può eseguire questa operazione utilizzando le autorizzazioni dell'entità chiamante, utilizzando un ruolo di servizio o utilizzando un ruolo collegato al servizio.
- **Sessioni di accesso diretto (FAS):** quando utilizzi un utente o un ruolo IAM per eseguire azioni AWS, sei considerato un principale. Quando si utilizzano alcuni servizi, è possibile eseguire un'operazione che attiva un'altra operazione in un servizio diverso. FAS utilizza le autorizzazioni del principale che chiama un Servizio AWS, combinate con la richiesta Servizio AWS per effettuare richieste ai servizi downstream. Le richieste FAS vengono effettuate solo quando un servizio riceve una richiesta che richiede interazioni con altri Servizi AWS o risorse per essere completata. In questo caso è necessario disporre delle autorizzazioni per eseguire entrambe le azioni. Per i dettagli delle policy relative alle richieste FAS, consulta la pagina [Forward access sessions](#).
- **Ruolo di servizio:** un ruolo di servizio è un [ruolo IAM](#) che un servizio assume per eseguire azioni per tuo conto. Un amministratore IAM può creare, modificare ed eliminare un ruolo di servizio dall'interno di IAM. Per ulteriori informazioni, consulta la sezione [Creazione di un ruolo per delegare le autorizzazioni a un Servizio AWS](#) nella Guida per l'utente di IAM.
- **Ruolo collegato al servizio:** un ruolo collegato al servizio è un tipo di ruolo di servizio collegato a un Servizio AWS. Il servizio può assumere il ruolo per eseguire un'operazione per tuo conto. I ruoli collegati al servizio vengono visualizzati nel tuo account Account AWS e sono di proprietà del servizio. Un amministratore IAM può visualizzare le autorizzazioni per i ruoli collegati ai servizi, ma non modificarle.

- Applicazioni in esecuzione su Amazon EC2: puoi utilizzare un ruolo IAM per gestire le credenziali temporanee per le applicazioni in esecuzione su un'istanza EC2 e che AWS CLI effettuano richieste API. AWS Ciò è preferibile all'archiviazione delle chiavi di accesso nell'istanza EC2. Per assegnare un AWS ruolo a un'istanza EC2 e renderlo disponibile per tutte le sue applicazioni, crei un profilo di istanza collegato all'istanza. Un profilo dell'istanza contiene il ruolo e consente ai programmi in esecuzione sull'istanza EC2 di ottenere le credenziali temporanee. Per ulteriori informazioni, consulta [Utilizzo di un ruolo IAM per concedere autorizzazioni ad applicazioni in esecuzione su istanze di Amazon EC2](#) nella Guida per l'utente di IAM.

Per informazioni sull'utilizzo dei ruoli IAM, consulta [Quando creare un ruolo IAM \(invece di un utente\)](#) nella Guida per l'utente di IAM.

## Gestione dell'accesso con policy

Puoi controllare l'accesso AWS creando policy e collegandole a AWS identità o risorse. Una policy è un oggetto AWS che, se associato a un'identità o a una risorsa, ne definisce le autorizzazioni. AWS valuta queste politiche quando un principale (utente, utente root o sessione di ruolo) effettua una richiesta. Le autorizzazioni nelle policy determinano l'approvazione o il rifiuto della richiesta. La maggior parte delle politiche viene archiviata AWS come documenti JSON. Per ulteriori informazioni sulla struttura e sui contenuti dei documenti delle policy JSON, consulta [Panoramica delle policy JSON](#) nella Guida per l'utente di IAM.

Gli amministratori possono utilizzare le policy AWS JSON per specificare chi ha accesso a cosa. In altre parole, quale principale può eseguire azioni su quali risorse e in quali condizioni.

Per impostazione predefinita, utenti e ruoli non dispongono di autorizzazioni. Per concedere agli utenti l'autorizzazione a eseguire azioni sulle risorse di cui hanno bisogno, un amministratore IAM può creare policy IAM. Successivamente l'amministratore può aggiungere le policy IAM ai ruoli e gli utenti possono assumere i ruoli.

Le policy IAM definiscono le autorizzazioni relative a un'azione, a prescindere dal metodo utilizzato per eseguirla. Ad esempio, supponiamo di disporre di una policy che consente l'azione `iam:GetRole`. Un utente con tale policy può ottenere informazioni sul ruolo dall' AWS Management Console AWS CLI, dall'API AWS API.

## Policy basate su identità

Le policy basate su identità sono documenti di policy di autorizzazione JSON che è possibile allegare a un'identità (utente, gruppo di utenti o ruoli IAM). Tali policy definiscono le azioni che utenti e ruoli



possono eseguire, su quali risorse e in quali condizioni. Per informazioni su come creare una policy basata su identità, consulta [Creazione di policy IAM](#) nella Guida per l'utente di IAM.

Le policy basate su identità possono essere ulteriormente classificate come policy inline o policy gestite. Le policy inline sono integrate direttamente in un singolo utente, gruppo o ruolo. Le politiche gestite sono politiche autonome che puoi allegare a più utenti, gruppi e ruoli nel tuo Account AWS. Le politiche gestite includono politiche AWS gestite e politiche gestite dai clienti. Per informazioni su come scegliere tra una policy gestita o una policy inline, consulta [Scelta fra policy gestite e policy inline](#) nella Guida per l'utente di IAM.

## Policy basate su risorse

Le policy basate su risorse sono documenti di policy JSON che è possibile collegare a una risorsa. Gli esempi più comuni di policy basate su risorse sono le policy di attendibilità dei ruoli IAM e le policy dei bucket Amazon S3. Nei servizi che supportano policy basate sulle risorse, gli amministratori dei servizi possono utilizzarle per controllare l'accesso a una risorsa specifica. Quando è collegata a una risorsa, una policy definisce le azioni che un principale può eseguire su tale risorsa e a quali condizioni. È necessario [specificare un principale](#) in una policy basata sulle risorse. I principali possono includere account, utenti, ruoli, utenti federati o. Servizi AWS

Le policy basate sulle risorse sono policy inline che si trovano in tale servizio. Non puoi utilizzare le policy AWS gestite di IAM in una policy basata sulle risorse.

## Liste di controllo degli accessi (ACL)

Le liste di controllo degli accessi (ACL) controllano quali principali (membri, utenti o ruoli dell'account) hanno le autorizzazioni per accedere a una risorsa. Le ACL sono simili alle policy basate su risorse, sebbene non utilizzino il formato del documento di policy JSON.

Amazon S3 e Amazon VPC sono esempi di servizi che supportano gli ACL. AWS WAF Per maggiori informazioni sulle ACL, consulta [Panoramica delle liste di controllo degli accessi \(ACL\)](#) nella Guida per gli sviluppatori di Amazon Simple Storage Service.

## Altri tipi di policy

AWS supporta tipi di policy aggiuntivi e meno comuni. Questi tipi di policy possono impostare il numero massimo di autorizzazioni concesse dai tipi di policy più comuni.

- Limiti delle autorizzazioni: un limite delle autorizzazioni è una funzione avanzata nella quale si imposta il numero massimo di autorizzazioni che una policy basata su identità può concedere a

un'entità IAM (utente o ruolo IAM). È possibile impostare un limite delle autorizzazioni per un'entità. Le autorizzazioni risultanti sono l'intersezione delle policy basate su identità dell'entità e i relativi limiti delle autorizzazioni. Le policy basate su risorse che specificano l'utente o il ruolo nel campo `Principal` sono condizionate dal limite delle autorizzazioni. Un rifiuto esplicito in una qualsiasi di queste policy sostituisce l'autorizzazione. Per ulteriori informazioni sui limiti delle autorizzazioni, consulta [Limiti delle autorizzazioni per le entità IAM](#) nella Guida per l'utente di IAM.

- Politiche di controllo dei servizi (SCP): le SCP sono politiche JSON che specificano le autorizzazioni massime per un'organizzazione o un'unità organizzativa (OU) in AWS Organizations. AWS Organizations è un servizio per il raggruppamento e la gestione centralizzata di più Account AWS di proprietà dell'azienda. Se abiliti tutte le funzionalità in un'organizzazione, puoi applicare le policy di controllo dei servizi (SCP) a uno o tutti i tuoi account. L'SCP limita le autorizzazioni per le entità negli account dei membri, inclusa ciascuna. Utente root dell'account AWS Per ulteriori informazioni su organizzazioni e policy SCP, consulta la pagina sulle [Policy di controllo dei servizi](#) nella Guida per l'utente di AWS Organizations .
- Policy di sessione: le policy di sessione sono policy avanzate che vengono trasmesse come parametro quando si crea in modo programmatico una sessione temporanea per un ruolo o un utente federato. Le autorizzazioni della sessione risultante sono l'intersezione delle policy basate su identità del ruolo o dell'utente e le policy di sessione. Le autorizzazioni possono anche provenire da una policy basata su risorse. Un rifiuto esplicito in una qualsiasi di queste policy sostituisce l'autorizzazione. Per ulteriori informazioni, consulta [Policy di sessione](#) nella Guida per l'utente di IAM.

## Più tipi di policy

Quando più tipi di policy si applicano a una richiesta, le autorizzazioni risultanti sono più complicate da comprendere. Per scoprire come si AWS determina se consentire una richiesta quando sono coinvolti più tipi di policy, consulta [Logica di valutazione delle policy](#) nella IAM User Guide.

## Come Servizi AWS lavorare con IAM

Per avere una visione di alto livello di come Servizi AWS funziona la maggior parte delle funzionalità IAM, consulta [AWS i servizi che funzionano con IAM nella IAM](#) User Guide.

Per scoprire come utilizzare uno specifico Servizio AWS con IAM, consulta la sezione sulla sicurezza della Guida per l'utente del servizio pertinente.

## Risoluzione dei problemi di AWS identità e accesso

Utilizza le seguenti informazioni per aiutarti a diagnosticare e risolvere i problemi più comuni che potresti riscontrare quando lavori con un AWS IAM.

### Argomenti

- [Non sono autorizzato a eseguire alcuna azione in AWS](#)
- [Non sono autorizzato a eseguire iam: PassRole](#)
- [Voglio consentire a persone esterne a me di accedere Account AWS alle mie AWS risorse](#)

### Non sono autorizzato a eseguire alcuna azione in AWS

Se ricevi un errore che indica che non sei autorizzato a eseguire un'operazione, le tue policy devono essere aggiornate per poter eseguire l'operazione.

L'errore di esempio seguente si verifica quando l'utente IAM `mateojackson` prova a utilizzare la console per visualizzare i dettagli relativi a una risorsa `my-example-widget` fittizia ma non dispone di autorizzazioni `aws:GetWidget` fittizie.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
aws:GetWidget on resource: my-example-widget
```

In questo caso, la policy per l'utente `mateojackson` deve essere aggiornata per consentire l'accesso alla risorsa `my-example-widget` utilizzando l'azione `aws:GetWidget`.

Se hai bisogno di aiuto, contatta il tuo AWS amministratore. L'amministratore è la persona che ti ha fornito le credenziali di accesso.

### Non sono autorizzato a eseguire iam: PassRole

Se ricevi un errore che indica che non sei autorizzato a eseguire l'operazione `iam:PassRole`, le tue policy devono essere aggiornate per poter passare un ruolo a AWS.

Alcuni Servizi AWS consentono di passare un ruolo esistente a quel servizio invece di creare un nuovo ruolo di servizio o un ruolo collegato al servizio. Per eseguire questa operazione, è necessario disporre delle autorizzazioni per trasmettere il ruolo al servizio.

L'errore di esempio seguente si verifica quando un utente IAM denominato `marymajor` cerca di utilizzare la console per eseguire un'operazione in AWS. Tuttavia, l'operazione richiede che il servizio

disponga delle autorizzazioni concesse da un ruolo di servizio. Mary non dispone delle autorizzazioni per passare il ruolo al servizio.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
iam:PassRole
```

In questo caso, le policy di Mary devono essere aggiornate per poter eseguire l'operazione `iam:PassRole`.

Se hai bisogno di aiuto, contatta il tuo AWS amministratore. L'amministratore è la persona che ti ha fornito le credenziali di accesso.

## Voglio consentire a persone esterne a me di accedere Account AWS alle mie AWS risorse

È possibile creare un ruolo con il quale utenti in altri account o persone esterne all'organizzazione possono accedere alle tue risorse. È possibile specificare chi è attendibile per l'assunzione del ruolo. Per servizi che supportano policy basate su risorse o liste di controllo accessi (ACL), utilizza tali policy per concedere alle persone l'accesso alle tue risorse.

Per ulteriori informazioni, consulta gli argomenti seguenti:

- Per sapere se AWS supporta queste funzionalità, consulta [Come Servizi AWS lavorare con IAM](#).
- Per scoprire come fornire l'accesso alle tue risorse attraverso Account AWS le risorse di tua proprietà, consulta [Fornire l'accesso a un utente IAM in un altro Account AWS di tua proprietà](#) nella IAM User Guide.
- Per scoprire come fornire l'accesso alle tue risorse a terze parti Account AWS, consulta [Fornire l'accesso a soggetti Account AWS di proprietà di terze parti](#) nella Guida per l'utente IAM.
- Per informazioni su come fornire l'accesso tramite la federazione delle identità, consulta [Fornire l'accesso a utenti autenticati esternamente \(Federazione delle identità\)](#) nella Guida per l'utente di IAM.
- Per informazioni sulle differenze tra l'utilizzo di ruoli e policy basate su risorse per l'accesso multi-account, consulta [Differenza tra i ruoli IAM e le policy basate su risorse](#) nella Guida per l'utente IAM.

# Convalida della conformità per questo AWS prodotto o servizio

Per sapere se un Servizio AWS programma rientra nell'ambito di specifici programmi di conformità, consulta Servizi AWS la sezione [Scope by Compliance Program Servizi AWS](#) e scegli il programma di conformità che ti interessa. Per informazioni generali, consulta Programmi di [AWS conformità Programmi](#) di di .

È possibile scaricare report di audit di terze parti utilizzando AWS Artifact. Per ulteriori informazioni, consulta [Scaricamento dei report in AWS Artifact](#) .

La vostra responsabilità di conformità durante l'utilizzo Servizi AWS è determinata dalla sensibilità dei dati, dagli obiettivi di conformità dell'azienda e dalle leggi e dai regolamenti applicabili. AWS fornisce le seguenti risorse per contribuire alla conformità:

- [Guide introduttive su sicurezza e conformità](#): queste guide all'implementazione illustrano considerazioni sull'architettura e forniscono passaggi per implementare ambienti di base incentrati sulla AWS sicurezza e la conformità.
- [Progettazione per la sicurezza e la conformità HIPAA su Amazon Web Services](#): questo white paper descrive in che modo le aziende possono utilizzare AWS per creare applicazioni idonee all'HIPAA.

## Note

Non Servizi AWS tutte sono idonee all'HIPAA. Per ulteriori informazioni, consulta la sezione [Riferimenti sui servizi conformi ai requisiti HIPAA](#).

- [AWS Risorse per la per la conformità](#): questa raccolta di cartelle di lavoro e guide potrebbe essere valida per il tuo settore e la tua località.
- [AWS Guide alla conformità dei clienti](#): comprendi il modello di responsabilità condivisa attraverso la lente della conformità. Le guide riassumono le migliori pratiche per la protezione Servizi AWS e mappano le linee guida per i controlli di sicurezza su più framework (tra cui il National Institute of Standards and Technology (NIST), il Payment Card Industry Security Standards Council (PCI) e l'International Organization for Standardization (ISO)).
- [Valutazione delle risorse con regole](#) nella Guida per gli AWS Config sviluppatori: il AWS Config servizio valuta la conformità delle configurazioni delle risorse alle pratiche interne, alle linee guida e alle normative del settore.

- [AWS Security Hub](#)— Ciò Servizio AWS fornisce una visione completa dello stato di sicurezza interno. AWS La Centrale di sicurezza utilizza i controlli di sicurezza per valutare le risorse AWS e verificare la conformità agli standard e alle best practice del settore della sicurezza. Per un elenco dei servizi e dei controlli supportati, consulta la pagina [Documentazione di riferimento sui controlli della Centrale di sicurezza](#).
- [Amazon GuardDuty](#): Servizio AWS rileva potenziali minacce ai tuoi carichi di lavoro Account AWS, ai contenitori e ai dati monitorando l'ambiente alla ricerca di attività sospette e dannose. GuardDuty può aiutarti a soddisfare vari requisiti di conformità, come lo standard PCI DSS, soddisfacendo i requisiti di rilevamento delle intrusioni imposti da determinati framework di conformità.
- [AWS Audit Manager](#)— Ciò Servizio AWS consente di verificare continuamente l' AWS utilizzo per semplificare la gestione del rischio e la conformità alle normative e agli standard di settore.

Questo AWS prodotto o servizio segue il [modello di responsabilità condivisa](#) attraverso i servizi specifici di Amazon Web Services (AWS) che supporta. Per informazioni sulla sicurezza dei AWS servizi, consulta la [pagina della documentazione sulla sicurezza del AWS servizio](#) e [AWS i servizi che rientrano nell'ambito delle iniziative di AWS conformità previste dal programma di conformità](#).

## Resilienza per questo AWS prodotto o servizio

L'infrastruttura AWS globale è costruita attorno a zone Regioni AWS di disponibilità.

Regioni AWS forniscono più zone di disponibilità fisicamente separate e isolate, collegate con reti a bassa latenza, ad alto throughput e altamente ridondanti.

Con le zone di disponibilità, puoi progettare e gestire applicazioni e database che eseguono automaticamente il failover tra zone di disponibilità senza interruzioni. Le zone di disponibilità sono più disponibili, tolleranti ai guasti e scalabili rispetto alle infrastrutture a data center singolo o multiplo tradizionali.

[Per ulteriori informazioni su AWS regioni e zone di disponibilità, vedere Global Infrastructure.AWS](#)

Questo AWS prodotto o servizio segue il [modello di responsabilità condivisa](#) attraverso i servizi specifici di Amazon Web Services (AWS) che supporta. Per informazioni sulla sicurezza dei AWS servizi, consulta la [pagina della documentazione sulla sicurezza del AWS servizio](#) e [AWS i servizi che rientrano nell'ambito delle iniziative di AWS conformità previste dal programma di conformità](#).

## Sicurezza dell'infrastruttura per questo AWS prodotto o servizio

Questo AWS prodotto o servizio utilizza servizi gestiti ed è pertanto protetto dalla sicurezza di rete AWS globale. Per informazioni sui servizi AWS di sicurezza e su come AWS protegge l'infrastruttura, consulta [AWS Cloud Security](#). Per progettare il tuo AWS ambiente utilizzando le migliori pratiche per la sicurezza dell'infrastruttura, vedi [Infrastructure Protection](#) in Security Pillar AWS Well-Architected Framework.

Utilizzate chiamate API AWS pubblicate per accedere a questo AWS Prodotto o Servizio attraverso la rete. I client devono supportare quanto segue:

- Transport Layer Security (TLS). È richiesto TLS 1.2 ed è consigliato TLS 1.3.
- Suite di cifratura con Perfect Forward Secrecy (PFS), ad esempio Ephemeral Diffie-Hellman (DHE) o Elliptic Curve Ephemeral Diffie-Hellman (ECDHE). La maggior parte dei sistemi moderni, come Java 7 e versioni successive, supporta tali modalità.

Inoltre, le richieste devono essere firmate utilizzando un ID chiave di accesso e una chiave di accesso segreta associata a un principale IAM. O puoi utilizzare [AWS Security Token Service](#) (AWS STS) per generare credenziali di sicurezza temporanee per sottoscrivere le richieste.

Questo AWS prodotto o servizio segue il [modello di responsabilità condivisa](#) attraverso i servizi specifici di Amazon Web Services (AWS) che supporta. Per informazioni sulla sicurezza dei AWS servizi, consulta la [pagina della documentazione sulla sicurezza del AWS servizio](#) e [AWS i servizi che rientrano nell'ambito delle iniziative di AWS conformità previste dal programma di conformità](#).

# Migrare dalla versione 1.x alla 2.x del AWS SDK for Java

La AWS SDK for Java 2.x è un'importante riscrittura della base di codice 1.x basata su Java 8+. Include numerosi aggiornamenti, tra cui maggiore consistenza, semplicità d'uso e immutabilità rigorosamente applicata. Questa sezione descrive le principali funzionalità nuove della versione 2.x e fornisce indicazioni su come migrare il codice alla versione 2.x dalla 1.x.

## Argomenti

- [Novità della versione 2](#)
- [step-by-step Istruzioni di migrazione con esempio](#)
- [Nome del pacchetto per le mappature ArtifactID di Maven](#)
- [Cosa c'è di diverso tra AWS SDK for Java 1.x e 2.x](#)
- [Uso SDK for Java 1.x side-by-side](#)

## Novità della versione 2

- È possibile configurare i propri client HTTP. [Vedi Configurazione del trasporto HTTP.](#)
- I client asincroni offrono un supporto I/O non bloccante e restituiscono oggetti. `CompletableFuture` [Vedere Programmazione asincrona.](#)
- Le operazioni che restituiscono più pagine dispongono di risposte con paginazione automatica. In questo modo, puoi concentrare il codice su cosa fare con la risposta, senza dover controllare e visualizzare le pagine successive. Vedi [Paginazione.](#)
- Le prestazioni dell'SDK all'avvio delle AWS Lambda funzioni sono migliorate. Vedi i miglioramenti delle [prestazioni dell'ora di avvio dell'SDK.](#)
- La versione 2.x supporta un nuovo metodo breve per la creazione delle richieste.

## Example

```
dynamoDbClient.putItem(request -> request.tableName(TABLE))
```

Per maggiori dettagli sulle nuove funzionalità e per vedere esempi di codice specifici, consulta le altre sezioni di questa guida.

- [Quick Start](#)



- [Configurazione](#)
- [Esempi di codice per la versione AWS SDK for Java 2.x](#)
- [Usa l'SDK](#)
- [Sicurezza per AWS SDK for Java](#)

## step-by-step Istruzioni di migrazione con esempio

Questa sezione fornisce una step-by-step guida per migrare l'applicazione che attualmente utilizza l'SDK for Java v1.x all'SDK for Java 2.x. La prima parte presenta una panoramica dei passaggi seguiti da un esempio dettagliato di migrazione.

I passaggi qui descritti descrivono la migrazione di un caso d'uso normale, in cui l'applicazione chiama Servizi AWS utilizzando client di servizi basati su modelli. Se devi migrare codice che utilizza API di livello superiore come [S3 Transfer Manager](#) o la [CloudFrontprefirma](#), consulta la sezione sotto il sommario. [the section called “Cosa c'è di diverso tra 1.x e 2.x”](#)

L'approccio qui descritto è un suggerimento. Puoi utilizzare altre tecniche e sfruttare le funzionalità di modifica del codice del tuo IDE per ottenere lo stesso risultato.

## Panoramica delle fasi

### 1. Inizia aggiungendo l'SDK for Java 2.x BOM

Aggiungendo l'elemento Maven BOM (Bill of Materials) per l'SDK for Java 2.x al file POM, ti assicuri che tutte le dipendenze v2 necessarie provengano dalla stessa versione. Il tuo POM può contenere dipendenze v1 e v2. Ciò consente di migrare il codice in modo incrementale anziché modificarlo tutto in una volta.

#### SDK per Java 2.x BOM

```
<dependencyManagement>
  <dependencies>
    <dependency>
      <groupId>software.amazon.awssdk</groupId>
      <artifactId>bom</artifactId>
      <version>2.24.3</version>
      <type>pom</type>
      <scope>import</scope>
```

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

Puoi trovare l'[ultima versione](#) sul Maven Central Repository.

## 2. Cerca nei file le istruzioni di importazione della classe v1

Scansionando i file dell'applicazione alla ricerca di Service\_ID utilizzati nelle importazioni v1, troverete i Service\_ID univoci utilizzati. Un SERVICE\_ID è un nome breve e univoco per un Servizio AWS. Ad esempio `cognitoidentity` è SERVICE\_ID per Amazon Cognito Identity.

## 3. Determina le dipendenze di Maven v2 dalle istruzioni di importazione v1

Dopo aver trovato tutti i Service\_ID v1 univoci, puoi determinare l'artefatto Maven corrispondente per la dipendenza v2 facendo riferimento a [the section called "Dal nome del pacchetto alle mappature ArtifactID"](#)

## 4. Aggiungere elementi di dipendenza v2 al file POM

Aggiorna il file Maven POM con gli elementi di dipendenza determinati nel passaggio 3.

## 5. Nei file Java, passa in modo incrementale dalle classi v1 alle classi v2

Quando sostituite le classi v1 con le classi v2, apportate le modifiche necessarie per supportare l'API v2, ad esempio utilizzando builder anziché costruttori e utilizzando getter e setter fluenti.

## 6. Rimuovi le dipendenze v1 Maven dalle importazioni POM e v1 dai file

Dopo aver migrato il codice per utilizzare le classi v2, rimuovi tutte le importazioni v1 rimanenti dai file e tutte le dipendenze dal file di build.

## 7. Rifattorizza il codice per utilizzare i miglioramenti dell'API v2

Dopo che il codice è stato compilato e superato con successo i test, puoi sfruttare i miglioramenti della v2, come l'utilizzo di un client HTTP o di impaginatori diversi per semplificare il codice. Questa fase è opzionale.

## Esempio di migrazione

In questo esempio, eseguiamo la migrazione di un'applicazione che utilizza l'SDK for Java v1 e accede a diversi Servizi AWS. Esaminiamo in dettaglio il seguente metodo v1 nel passaggio 5.

Questo è un metodo in una classe che contiene otto metodi e nell'applicazione sono presenti 32 classi.

metodo v1 per migrare

Di seguito sono elencate solo le importazioni dell'SDK v1 dal file Java.

```
import com.amazonaws.ClientConfiguration;
import com.amazonaws.regions.Region;
import com.amazonaws.regions.RegionUtils;
import com.amazonaws.services.ec2.AmazonEC2Client;
import com.amazonaws.services.ec2.model.AmazonEC2Exception;
import com.amazonaws.services.ec2.model.CreateTagsRequest;
import com.amazonaws.services.ec2.model.DescribeInstancesRequest;
import com.amazonaws.services.ec2.model.DescribeInstancesResult;
import com.amazonaws.services.ec2.model.Instance;
import com.amazonaws.services.ec2.model.InstanceStateName;
import com.amazonaws.services.ec2.model.Reservation;
import com.amazonaws.services.ec2.model.Tag;
import com.amazonaws.services.ec2.model.TerminateInstancesRequest;
...
private static List<Instance> getRunningInstances(AmazonEC2Client ec2, List<String>
instanceIds) {
    List<Instance> runningInstances = new ArrayList<>();
    try {
        DescribeInstancesRequest request = new DescribeInstancesRequest()
            .withInstanceIds(instanceIds);
        DescribeInstancesResult result;
        do {
            // DescribeInstancesResponse is a paginated response, so use tokens with
multiple requests.
            result = ec2.describeInstances(request);
            request.setNextToken(result.getNextToken()); // Prepare request for next
page.
            for (final Reservation r : result.getReservations()) {
                for (final Instance instance : r.getInstances()) {
                    LOGGER.info("Examining instanceId: "+ instance.getInstanceId());
                    // if instance is in a running state, add it to runningInstances
list.
                    if (RUNNING_STATES.contains(instance.getState().getName())) {
                        runningInstances.add(instance);
                    }
                }
            }
        }
    }
}
```

```

    } while (result.getNextToken() != null);
} catch (final AmazonEC2Exception exception) {
    // if instance isn't found, assume its terminated and continue.
    if (exception.getErrorCode().equals(NOT_FOUND_ERROR_CODE)) {
        LOGGER.info("Instance probably terminated; moving on.");
    } else {
        throw exception;
    }
}
return runningInstances;
}

```

## 1. Aggiungi v2 Maven BOM

Aggiungi la BOM Maven per l'SDK for Java 2.x al POM insieme a tutte le altre dipendenze nella sezione. `dependencyManagement`. Se il tuo file POM contiene la BOM per la versione 1 dell'SDK, lasciala per ora. Verrà rimosso in una fase successiva.

### Gestione delle dipendenze POM fin dall'inizio

```

<dependencyManagement>
  <dependencies>
    <dependency>
      <groupId>org.example</groupId>           <!--Existing dependency in POM. -->
      <artifactId>bom</artifactId>
      <version>1.3.4</version>
      <type>pom</type>
      <scope>import</scope>
    </dependency>
    ...
    <dependency>
      <groupId>com.amazonaws</groupId>
      <artifactId>aws-java-sdk-bom</artifactId> <!--Existing v1 BOM dependency. -->
      <version>1.11.1000</version>
      <type>pom</type>
      <scope>import</scope>
    </dependency>
    ...
    <dependency>
      <groupId>software.amazon.awssdk</groupId> <!--Add v2 BOM dependency. -->
      <artifactId>bom</artifactId>
      <version>2.24.3</version>
      <type>pom</type>

```

```
<scope>import</scope>
</dependency>
</dependencies>
</dependencyManagement>
```

## 2. Cerca nei file le istruzioni di importazione della classe v1

Cerca nel codice dell'applicazione le occorrenze uniche di `import com.amazonaws.services`. Questo ci aiuta a determinare le dipendenze v1 utilizzate dal progetto. Se la tua applicazione ha un file Maven POM con le dipendenze v1 elencate, puoi invece utilizzare queste informazioni.

Per questo esempio utilizziamo il comando [ripgrep\(rg\)](#) per cercare nella base di codice.

Dalla radice della tua base di codice, esegui il seguente `ripgrep` comando. Dopo aver `ripgrep` trovato le istruzioni di importazione, queste vengono reindirizzate a `cut`, `sort`, e `uniq` i comandi per isolare i `Service_ID`.

```
rg --no-filename 'import\s+com\.amazonaws\.services' | cut -d '.' -f 4 | sort | uniq
```

Per questa applicazione, i seguenti `Service_ID` vengono registrati nella console.

```
autoscaling
cloudformation
ec2
identitymanagement
```

Ciò indica che c'è stata almeno un'occorrenza di ciascuno dei seguenti nomi di pacchetto utilizzati nelle istruzioni `import`. Per i nostri scopi, i nomi delle singole classi non contano. Abbiamo solo bisogno di trovare i `service_ID` utilizzati.

```
com.amazonaws.services.autoscaling.*
com.amazonaws.services.cloudformation.*
com.amazonaws.services.ec2.*
com.amazonaws.services.identitymanagement.*
```

## 3. Determina le dipendenze di Maven v2 dalle istruzioni di importazione v1

I `Service_ID` per la v1 che abbiamo isolato dalla Fase 2, ad esempio `autoscaling` e `cloudformation` possono essere mappati sullo stesso `SERVICE_ID` v2 per la maggior parte.

Poiché nella maggior parte dei casi l'ArtifactID di Maven v2 corrisponde al SERVICE\_ID, hai le informazioni necessarie per aggiungere blocchi di dipendenza al tuo file POM.

La tabella seguente mostra come possiamo determinare le dipendenze v2.

v1 SERVICE_ID è mappato a... nome del pacchetto	v2 SERVICE_ID mappa su... nome del pacchetto	dipendenza da Maven v2
ec2  com.amazonaws.services. <b>ec2</b> .*	ec2  software.amazon.awssdk.services. <b>ec2</b> .*	<pre>&lt;dependency&gt;   &lt;groupId&gt;software. amazon.awssdk&lt;/gro upId&gt;   &lt;artifactId&gt; <b>ec2</b>&lt;/ artifactId&gt; &lt;/dependency&gt;</pre>
scalabilità automatica  com.amazonaws.services. <b>autoscaling</b> .*	scalabilità automatica  software.amazon.awssdk.services. <b>autoscaling</b> .*	<pre>&lt;dependency&gt;   &lt;groupId&gt;software. amazon.awssdk&lt;/gro upId&gt;   &lt;artifactId&gt; <b>autoscali ng</b> &lt;/artifactId&gt; &lt;/dependency&gt;</pre>
cloudformation  com.amazonaws.services. <b>cloudform ation</b> .*	cloudformation  software.amazon.awssdk. <b>cloudform ation</b> .*	<pre>&lt;dependency&gt;   &lt;groupId&gt;software. amazon.awssdk&lt;/gro upId&gt;   &lt;artifactId&gt; <b>cloudform ation</b> &lt;/artifactId&gt; &lt;/dependency&gt;</pre>
gestione dell'identità*  com.amazonaws.services. <b>identitym anagement</b> .*	sono*  software.amazon.awssdk. <b>iam</b> .*	<pre>&lt;dependency&gt;   &lt;groupId&gt;software. amazon.awssdk&lt;/gro upId&gt;   &lt;artifactId&gt; <b>iam</b>&lt;/ artifactId&gt;</pre>

v1 SERVICE_ID è mappato a...	v2 SERVICE_ID mappa su... nome del pacchetto	dipendenza da Maven v2
nome del pacchetto		</dependency>

\* La iam mappatura identitymanagement to è un'eccezione in cui il SERVICE\_ID differisce tra le versioni. Fai riferimento a [the section called “Dal nome del pacchetto alle mappature ArtifactID”](#) per le eccezioni se Maven o Gradle non sono in grado di risolvere la dipendenza v2.

#### 4. Aggiungi elementi di dipendenza v2 al file POM

Nel passaggio 3, abbiamo determinato i quattro blocchi di dipendenza che devono essere aggiunti al file POM. Non è necessario aggiungere una versione perché abbiamo specificato la BOM nel passaggio 1. Dopo aver aggiunto le importazioni, il nostro file POM presenta i seguenti elementi di dipendenza.

```

...
<dependencies>
  ...
  <dependency>
    <groupId>software.amazon.awssdk</groupId>
    <artifactId>autoscaling</artifactId>
  </dependency>
  <dependency>
    <groupId>software.amazon.awssdk</groupId>
    <artifactId>iam</artifactId>
  </dependency>
  <dependency>
    <groupId>software.amazon.awssdk</groupId>
    <artifactId>cloudformation</artifactId>
  </dependency>
  <dependency>
    <groupId>software.amazon.awssdk</groupId>
    <artifactId>ec2</artifactId>
  </dependency>
  ...
</dependencies>
...

```

## 5. Nei file Java, passa in modo incrementale dalle classi v1 alle classi v2

Nel metodo che stiamo migrando, vediamo

- Un client di servizio EC2 di `com.amazonaws.services.ec2.AmazonEC2Client`
- Sono state utilizzate diverse classi di modelli EC2. Ad esempio `DescribeInstancesRequest` e `DescribeInstancesResult`

```
import com.amazonaws.ClientConfiguration;
import com.amazonaws.regions.Region;
import com.amazonaws.regions.RegionUtils;
import com.amazonaws.services.ec2.AmazonEC2Client;
import com.amazonaws.services.ec2.model.AmazonEC2Exception;
import com.amazonaws.services.ec2.model.CreateTagsRequest;
import com.amazonaws.services.ec2.model.DescribeInstancesRequest;
import com.amazonaws.services.ec2.model.DescribeInstancesResult;
import com.amazonaws.services.ec2.model.Instance;
import com.amazonaws.services.ec2.model.InstanceStateName;
import com.amazonaws.services.ec2.model.Reservation;
import com.amazonaws.services.ec2.model.Tag;
import com.amazonaws.services.ec2.model.TerminateInstancesRequest;
...
private static List<Instance> getRunningInstances(AmazonEC2Client ec2, List<String>
instanceIds)
    List<Instance> runningInstances = new ArrayList<>();
    try {
        DescribeInstancesRequest request = new DescribeInstancesRequest()
            .withInstanceIds(instanceIds);
        DescribeInstancesResult result;
        do {
            // DescribeInstancesResponse is a paginated response, so use tokens with
multiple re
            result = ec2.describeInstances(request);
            request.setNextToken(result.getNextToken()); // Prepare request for next
page.
            for (final Reservation r : result.getReservations()) {
                for (final Instance instance : r.getInstances()) {
                    LOGGER.info("Examining instanceId: "+ instance.getInstanceId());
                    // if instance is in a running state, add it to runningInstances
list.
                    if (RUNNING_STATES.contains(instance.getState().getName())) {
                        runningInstances.add(instance);
                    }
                }
            }
        } while (result.isTruncated());
    } catch (AmazonEC2Exception e) {
        LOGGER.error("Error while describing instances: " + e.getMessage());
    }
}
```



```
        }
    }
} while (result.getNextToken() != null);
} catch (final AmazonEC2Exception exception) {
    // if instance isn't found, assume its terminated and continue.
    if (exception.getErrorCode().equals(NOT_FOUND_ERROR_CODE)) {
        LOGGER.info("Instance probably terminated; moving on.");
    } else {
        throw exception;
    }
}
return runningInstances;
}
...

```

Il nostro obiettivo è sostituire tutte le importazioni v1 con le importazioni v2. Procediamo una lezione alla volta.

a. Sostituisci l'istruzione di importazione o il nome della classe

Vediamo che il primo parametro del `describeRunningInstances` metodo è un'AmazonEC2Client istanza v1. Esegui una di queste operazioni:

- Sostituisci l'importazione per `com.amazonaws.services.ec2.AmazonEC2Client` con `software.amazon.awssdk.services.ec2.Ec2Client` e passa `AmazonEC2Client` a `Ec2Client`.
- Cambia il tipo di parametro in `Ec2Client` e lascia che l'IDE ci richieda l'importazione corretta. Il nostro IDE ci chiederà di importare la classe v2 perché i nomi dei client sono diversi: e. `AmazonEC2Client` `Ec2Client` Questo approccio non funziona se il nome della classe è lo stesso in entrambe le versioni.

b. Sostituisci le classi del modello v1 con gli equivalenti v2

Dopo la modifica alla `v2Ec2Client`, se utilizziamo un IDE, vediamo errori di compilazione nella seguente dichiarazione.

```
result = ec2.describeInstances(request);
```

L'errore di compilazione deriva dall'utilizzo di un'istanza di v1 `DescribeInstancesRequest` come parametro del metodo v2. `Ec2Client describeInstances` Per risolvere il problema, effettuate le seguenti istruzioni di sostituzione o importazione.

replace	con
<pre>import com.amazonaws.services.ec2.model.DescribeInstancesRequest</pre>	<pre>import software.amazon.awssdk.services.ec2.model.DescribeInstancesRequest</pre>

c. Cambia i costruttori v1 in v2 builder.

Vediamo ancora errori di compilazione perché [non ci](#) sono costruttori nelle classi v2. Per risolvere il problema, apporta la seguente modifica.

modifica	in
<pre>final DescribeInstancesRequest request = new DescribeInstancesRequest().withInstanceIds(instanceIdsCopy);</pre>	<pre>final DescribeInstancesRequest request = DescribeInstancesRequest.builder().instanceIds(instanceIdsCopy).build();</pre>

d. Sostituisci gli oggetti di **\*Result** risposta v1 con gli equivalenti **\*Response** v2

Una differenza consistente tra v1 e v2 è che tutti [gli oggetti di risposta in v2](#) terminano con invece di. **\*Response** **\*Result** Sostituisci l'importazione v1 con `DescribeInstancesResult` l'importazione v2, `DescribeInstancesResponse`

d. Apporta modifiche all'API

La seguente dichiarazione richiede alcune modifiche.

```
request.setNextToken(result.getNextToken());
```

Nella v2, [i metodi setter](#) non utilizzano set o with. prefix I metodi Getter con prefisso get sono presenti anche nell'SDK for Java 2.x

Le classi del modello, come l'requestistanza, sono immutabili nella v2, quindi dobbiamo crearne una nuova con un builder. DescribeInstancesRequest

Nella v2, l'istruzione diventa la seguente.

```
request = DescribeInstancesRequest.builder()
    .nextToken(result.nextToken())
    .build();
```

d. Ripetere l'operazione fino alla compilazione del metodo con le classi v2

Continua con il resto del codice. Sostituisci le importazioni v1 con le importazioni v2 e correggi gli errori di compilazione. Se necessario, consulta il riferimento all'[API v2](#) e il riferimento [What's different](#).

Dopo aver migrato questo singolo metodo, abbiamo il seguente codice v2.

```
import com.amazonaws.ClientConfiguration;
import com.amazonaws.regions.Region;
import com.amazonaws.regions.RegionUtils;
import com.amazonaws.services.ec2.AmazonEC2Client;
import com.amazonaws.services.ec2.model.AmazonEC2Exception;
import com.amazonaws.services.ec2.model.CreateTagsRequest;
import com.amazonaws.services.ec2.model.InstanceStateName;
import com.amazonaws.services.ec2.model.Tag;
import com.amazonaws.services.ec2.model.TerminateInstancesRequest;

import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.DescribeInstancesRequest;
import software.amazon.awssdk.services.ec2.model.DescribeInstancesResponse;
import software.amazon.awssdk.services.ec2.model.Ec2Exception;
import software.amazon.awssdk.services.ec2.model.Instance;
import software.amazon.awssdk.services.ec2.model.Reservation;
...
private static List<Instance> getRunningInstances(Ec2Client ec2, List<String>
instanceIds) {
    List<Instance> runningInstances = new ArrayList<>();
    try {
        DescribeInstancesRequest request = DescribeInstancesRequest.builder()
            .instanceIds(instanceIds)
            .build();
```

```

        DescribeInstancesResponse result;
        do {
            // DescribeInstancesResponse is a paginated response, so use tokens
with multiple re
            result = ec2.describeInstances(request);
            request = DescribeInstancesRequest.builder() // Prepare request for
next page.
                .nextToken(result.nextToken())
                .build();
            for (final Reservation r : result.reservations()) {
                for (final Instance instance : r.instances()) {
                    // if instance is in a running state, add it to
runningInstances list.
                    if (RUNNING_STATES.contains(instance.state().nameAsString())) {
                        runningInstances.add(instance);
                    }
                }
            }
        } while (result.nextToken() != null);
    } catch (final Ec2Exception exception) {
        // if instance isn't found, assume its terminated and continue.
        if (exception.awsErrorDetails().errorCode().equals(NOT_FOUND_ERROR_CODE)) {
            LOGGER.info("Instance probably terminated; moving on.");
        } else {
            throw exception;
        }
    }
    return runningInstances;
}
...

```

Poiché stiamo migrando un singolo metodo in un file Java con otto metodi, abbiamo un mix di importazioni v1 e v2 mentre elaboriamo il file. Abbiamo aggiunto le ultime sei istruzioni di importazione man mano che eseguiamo i passaggi.

Dopo aver migrato tutto il codice, non ci saranno più istruzioni di importazione v1.

## 6. Rimuovi le dipendenze v1 Maven dalle importazioni POM e v1 dai file

Dopo aver migrato tutto il codice v1 nel file, abbiamo le seguenti istruzioni di importazione dell'SDK v2.

```
import software.amazon.awssdk.core.client.config.ClientOverrideConfiguration;
```

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.regions.ServiceMetadata;
import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.CreateTagsRequest;
import software.amazon.awssdk.services.ec2.model.DescribeInstancesRequest;
import software.amazon.awssdk.services.ec2.model.DescribeInstancesResponse;
import software.amazon.awssdk.services.ec2.model.Ec2Exception;
import software.amazon.awssdk.services.ec2.model.Instance;
import software.amazon.awssdk.services.ec2.model.InstanceStateName;
import software.amazon.awssdk.services.ec2.model.Reservation;
import software.amazon.awssdk.services.ec2.model.Tag;
import software.amazon.awssdk.services.ec2.model.TerminateInstancesRequest;
```

Dopo aver migrato tutti i file nella nostra applicazione, non abbiamo più bisogno delle dipendenze v1 nel nostro file POM. Rimuovi la BOM v1 dalla dependencyManagement sezione, se in uso, e tutti i blocchi di dipendenza v1.

## 7. Rifattorizza il codice per utilizzare i miglioramenti dell'API v2

Per lo snippet che stiamo migrando, possiamo opzionalmente utilizzare un paginatore v2 e lasciare che l'SDK gestisca le richieste basate su token per ulteriori dati.

Possiamo sostituire l'intera clausola con la seguente. do

```
DescribeInstancesIterable responses =
ec2.describeInstancesPaginator(request);

responses.reservations().stream()
    .forEach(reservation -> reservation.instances()
        .forEach(instance -> {
            if
(RUNNING_STATES.contains(instance.state().nameAsString())) {
                runningInstances.put(instance.instanceId(),
instance);
            }
        }));
```

## Nome del pacchetto per le mappature ArtifactID di Maven

Quando migri il tuo progetto Maven o Gradle dalla v1 dell'SDK for Java alla v2, devi capire quali dipendenze aggiungere al tuo file di build. L'approccio descritto nel [the section called "tep-by-step"](#)

[Istruzioni S](#)” (passaggio 3) utilizza i nomi dei pacchetti nelle istruzioni di importazione come punto di partenza per determinare le dipendenze (come ArtifactID) da aggiungere al file di build.

È possibile utilizzare le informazioni in questo argomento per mappare i nomi dei pacchetti v1 agli artifactID v2.

## Convenzione di denominazione comune utilizzata nei nomi dei pacchetti e nei Maven ArtifactID

La tabella seguente mostra la convenzione di denominazione comune utilizzata dagli SDK per un determinato SERVICE\_ID. Un SERVICE\_ID è un identificatore univoco per un Servizio AWS. Ad esempio, il SERVICE\_ID per il servizio Amazon S3 è s3 ed `cognitoidentity` è il SERVICE\_ID per Amazon Cognito Identity.

nome del pacchetto v1 (dichiarazione di importazione)	artifactID v1	artifactID v2	nome del pacchetto v2 (istruzione di importazione)
<code>com.amazonaws.services.service_id</code>	<code>aws-java-sdk-ID_SERVIZIO</code>	<code>ID_SERVIZIO</code>	<code>software.amazon.awssdk.services.service_id</code>

Esempio di Amazon Cognito Identity (SERVICE\_ID:) ***cognitoidentity***

<code>com.amazonaws.services.identità cognitiva</code>	<code>aws-java-sdk-identità cognitiva</code>	<code>identità cognitiva</code>	<code>software.amazon.awssdk.services.identità cognitiva</code>
--	--	---------------------------------	---

## differenze SERVICE\_ID

### All'interno della v1

In alcuni casi il SERVICE\_ID differisce tra il nome del pacchetto e l'artifactID per lo stesso servizio. Ad esempio, la riga CloudWatch Metrics della tabella seguente mostra che `metrics` è il SERVICE\_ID nel nome del pacchetto ma è il SERVICE\_ID dell'ArtifactID. `cloudwatchmetrics`

## All'interno della v2

Non ci sono differenze nel SERVICE\_ID utilizzato nei nomi dei pacchetti e negli ArtifactID.

## Tra v1 e v2

Per la maggior parte dei servizi, il SERVICE\_ID nella v2 è lo stesso del SERVICE\_ID della v1 sia nei nomi dei pacchetti che negli ArtifactID. Un esempio di ciò è il SERVICE\_ID, come mostrato nella tabella precedente. `cognitoedentity` Tuttavia, alcuni SERVICE\_ID differiscono tra gli SDK, come mostrato nella tabella seguente.

Un SERVICE\_ID in grassetto in una delle colonne v1 indica che è diverso dal SERVICE\_ID utilizzato nella v2.

Nome servizio	nome del pacchetto v1	artifactID v1	artifactID v2	nome del pacchetto v2
	Tutti i nomi dei pacchetti iniziano con <code>com.amazonaws.services</code> quello mostrato nella prima riga.	Tutti gli ArtifactID sono racchiusi tra tag, come mostrato nella prima riga.	Tutti gli ArtifactID sono racchiusi tra tag, come mostrato nella prima riga.	Tutti i nomi dei pacchetti iniziano con <code>software.amazon.awssdk</code> quello mostrato nella prima riga.
API Gateway	<code>com.amazonaws.services.apigateway</code>	<code>&lt;artifactId&gt;aws-java-sdk-gateway API&lt;/artifactId&gt;</code>	<code>&lt;artifactId&gt;una porta d'accesso&lt;/artifactId&gt;</code>	<code>software.amazon.awssdk.services.apigateway</code>
Registro delle app	registro delle app	apprendistato	registro del catalogo dei servizi	registro del catalogo dei servizi
Application Discovery	scoperta delle applicazioni	scoperta	scoperta delle applicazioni	individuazione delle applicazioni

Nome servizio	nome del pacchetto v1	artifactID v1	artifactID v2	nome del pacchetto v2
Runtime AI Augmented	autonomia aerea aumentata	durata dell'aria aumentata	runtime sagemaker a2i	runtime sagemakera2i
Certificate Manager	gestore dei certificati	acm	acm	acm
CloudControl API	API di controllo cloud	API di controllo cloud	controllo cloud	controllo del cloud
CloudSearch	cloudsearch v2	cloudsearch	cloudsearch	cloudsearch
CloudSearch Dominio	dominio cloudsearch	ricerca nel cloud	dominio cloudsearch	dominio cloudsearch
CloudWatch Eventi	eventi cloudwatch	eventi	eventi cloudwatch	eventi cloudwatch
CloudWatch Evidentemente	Cloudwatch evidentemente	cloudwatch evidentemente	evidently	evidently
CloudWatch Registri	logs	logs	log di cloudwatch	cloudwatchlogs
CloudWatch Metriche	metriche	metriche cloudwatch	cloudwatch	cloudwatch
CloudWatch rum	Cloudwatch rum	cloudwatchrum	rum	rum
Provider di identità Cognito	cognitoidp	cognitoidp	provider di cognitoidentity	provider di cognitoidentity
Campagna Connect	connetti la campagna	connetti la campagna	collegare le campagne	collegare le campagne
Connect Wisdom	connetti la saggezza	connettere la saggezza	wisdom	wisdom



Nome servizio	nome del pacchetto v1	artifactID v1	artifactID v2	nome del pacchetto v2
Database Migration Service	servizio di migrazione del database	dms	migrazione del database	migrazione del database
DataZone	zona dati	zona dati esterna	zona dati	zona dati
DynamoDB	dinamo dbv2	dynamodb	dynamodb	dynamodb
File system elastico	file system elastico	efs	efs	efs
Riduzione elastica della mappa	elasticmapreduce	emr	emr	emr
Glue DataBrew	incolla databrew	incollare databrew	databrew	databrew
IAM Roles Anywhere	sono ruoli ovunque	sono ruoli ovunque	rolesanywhere	rolesanywhere
Gestione di identità	gestione dell'identità	iam	iam	iam
Dati IoT	dati IoT	iot	piano dati iot	piano dati iot
Analisi Kinesis	kinesisanalytics	kinesis	kinesisanalytics	kinesisanalytics
Kinesis Firehose	manichetta antincendio kinesis	kinesis	firehose	firehose
Canali di segnalazione video Kinesis	canali di segnalazione video kinesis	canali di segnalazione video kinesis	segnalazione video kinesis	segnalazione video kinesis
Lex	runtime lex	lex	runtime lex	lexruntime

Nome servizio	nome del pacchetto v1	artifactID v1	artifactID v2	nome del pacchetto v2
Attenzione a Vision	fai attenzione alla visione	attenzione alla visione	lookoutvision	lookoutvision
Modernizzazione del mainframe	modernizzazione del mainframe	modernizzazione del mainframe	m2	m2
Misurazione del Marketplace	misurazione del mercato	servizio di marketpla cemetering	misurazione del marketplace	misurazione del marketplace
Grafana gestita	grafana gestito	grafana gestito	grafana	grafana
Mechanical Turk	mturk	richiedente turco meccanico	mturk	mturk
Suggerimenti di strategia dell'Hub di migrazione	raccomand azioni sulla strategia dell'hub di migrazione	raccomandazioni strategiche di migrationhub	strategia migrationhub	strategia migrationhub
Nimble Studio	studio agile	studio agile	nimble	nimble
5G privato	5g privato	5g privato	reti private	reti private
Prometheus	prometeo	prometeo	amp	amp
Cestino	bidone per riciclare	pattumiera	rbin	rbin
API dati di Redshift	redshift data api	redshiftdataapi	dati redshift	dati redshift
Route 53	domini route53	route53	domini route53	domini route53
Sage Maker Edge Manager	sagemaker edgemanager	sagemaked gemanager	sagemakeredge	sagemaker edge

Nome servizio	nome del pacchetto v1	artifactID v1	artifactID v2	nome del pacchetto v2
Token di sicurezza	token di sicurezza	sts	sts	sts
Migrazione dei server	migrazione del server	migrazione del server	sms	sms
Email semplice	email semplice	ses	ses	ses
Email semplice V2	email semplice v2	ses v2	sesv2	sesv2
Gestione semplice dei sistemi	gestione semplice dei sistemi	ssm	ssm	ssm
Flusso di lavoro semplice	flusso di lavoro semplice	flusso di lavoro semplice	swf	swf
Step Functions	funzioni a fasi	funzioni step	sfn	sfn

## Cosa c'è di diverso tra AWS SDK for Java 1.x e 2.x

Questa sezione descrive le principali modifiche da tenere a mente quando si converte un'applicazione dall'utilizzo della AWS SDK for Java versione 1.x alla versione 2.x.

### Modifica del nome del pacchetto

Una modifica notevole rispetto all'SDK for Java 1.x all'SDK for Java 2.x è la modifica del nome del pacchetto. I nomi dei pacchetti iniziano con `software.amazon.awssdk` SDK 2.x, mentre lo utilizza l'SDK 1.x. `com.amazonaws`

Questi stessi nomi differenziano gli artefatti Maven da SDK 1.x a SDK 2.x. Gli artefatti Maven per l'SDK 2.x utilizzano GroupID, mentre l'SDK 1.x utilizza `software.amazon.awssdk` GroupID. `com.amazonaws`

In alcuni casi il codice richiede una dipendenza per un progetto che altrimenti utilizza solo artefatti SDK 2.x `com.amazonaws`. Un esempio di ciò è quando si lavora con il lato server. AWS Lambda. Questo è stato mostrato nella sezione [Configurazione di un progetto Apache Maven precedente](#) di questa guida.

### Note

Diversi nomi di pacchetto nell'SDK 1.x contengono `v2`. L'uso di `v2` in questo caso di solito significa che il codice contenuto nel pacchetto è destinato a funzionare con la versione 2 del servizio.

Poiché il nome completo del pacchetto inizia con `com.amazonaws`, si tratta di componenti SDK 1.x. Esempi di questi nomi di pacchetto nell'SDK 1.x sono:

- `com.amazonaws.services.dynamodbv2`
- `com.amazonaws.retry.v2`
- `com.amazonaws.services.apigatewayv2`
- `com.amazonaws.services.simpleemailv2`

## Aggiungere la versione 2.x al progetto

Maven è il modo consigliato per gestire le dipendenze quando si utilizza la 2.x. AWS SDK for Java. Per aggiungere componenti della versione 2.x al progetto, aggiorna il `pom.xml` file con una dipendenza dall'SDK.

### Example

```
<dependencyManagement>
  <dependencies>
    <dependency>
      <groupId>software.amazon.awssdk</groupId>
      <artifactId>bom</artifactId>
      <version>2.16.1</version>
      <type>pom</type>
      <scope>import</scope>
    </dependency>
  </dependencies>
</dependencyManagement>

<dependencies>
```

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

Puoi anche [utilizzare le versioni 1.x e 2.x side-by-side durante](#) la migrazione del progetto alla versione 2.x.

## POJO non modificabili

Client, richieste di operazione e oggetti di risposta non sono più modificabili e non possono essere cambiati dopo la creazione. Per riutilizzare una variabile di richiesta o di risposta, è necessario creare un nuovo oggetto per assegnarla.

Example di aggiornamento di un oggetto richiesta in 1.x

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

request.setNextToken(response.getNextToken());
```

Example di aggiornamento di un oggetto richiesta in 2.x

```
DescribeAlarmsRequest request = DescribeAlarmsRequest.builder().build();
DescribeAlarmsResponse response = cw.describeAlarms(request);

request = DescribeAlarmsRequest.builder()
    .nextToken(response.getNextToken())
    .build();
```

## Metodi Setter e Getter

Nella versione AWS SDK for Java 2.x, i nomi dei metodi setter non includono il prefisso `or. set with`. Ad esempio, `*.withEndpoint()` è adesso `*.endpoint()`

I nomi dei metodi Getter non utilizzano il `get` prefisso.

Example dell'utilizzo dei metodi setter in 1.x

```
AmazonDynamoDB client = AmazonDynamoDBClientBuilder.standard()
```

```
.withRegion("us-east-1")  
.build();
```

Example dell'utilizzo dei metodi setter in 2.x

```
DynamoDbClient client = DynamoDbClient.builder()  
.region(Region.US_EAST_1)  
.build();
```

Example di usare i metodi getter in 1.x

```
String token = request.getNextToken();
```

Example di usare i metodi getter in 2.x

```
String token = request.nextToken();
```

## nomi delle classi del modello

I nomi delle classi del modello che rappresentano le risposte di servizio terminano con `Response` la v2 anziché con `Result` quella utilizzata dalla v1.

Example di nomi di classi che rappresentano una risposta in v1

```
CreateApiKeyResult  
AllocateAddressResult
```

Example di nomi di classi che rappresentano una risposta in v2

```
CreateApiKeyResponse  
AllocateAddressResponse
```

## Stato della migrazione di librerie e utilità

### SDK for Java: librerie e utilità

La tabella seguente elenca lo stato di migrazione delle librerie e delle utilità per l'SDK for Java.

Nome della versione 1.12.x	Nome della versione 2.x	Dalla versione 2.x
DynamoDBMapper	<a href="#">DynamoDbEnhancedClient</a>	2.12.0
Waiter	<a href="#">Waiter</a>	2.15.0
CloudFrontUrlSigner, CloudFrontCookieSigner	<a href="#">CloudFrontUtilities</a>	2.18.33
TransferManager	<a href="#">S3 TransferManager</a>	2.19.0
Client EC2 Metadati	<a href="#">Client EC2 Metadata</a>	2.19.29
Analizzatore URI S3	<a href="#">Analizzatore URI S3</a>	2.20.41
IAM Policy Builder	<a href="#">IAM Policy Builder</a>	2.20.126
Buffering lato client di Amazon SQS	Raggruppamento automatico delle richieste	<a href="#">non ancora rilasciato</a>
Listener di avanzamento	Listener di avanzamento	<a href="#">non ancora rilasciato</a>

## librerie correlate

La tabella seguente elenca le librerie rilasciate separatamente ma che funzionano con l'SDK for Java 2.x.

Nome utilizzato con la versione 2.x dell'SDK for Java	Dalla versione
<a href="#">Client di crittografia Amazon S3</a>	3.0.0 1
<a href="#">AWS Client di crittografia del database per DynamoDB</a>	3.0.0 2

<sup>1</sup> Il client di crittografia per Amazon S3 è disponibile utilizzando la seguente dipendenza Maven.

```
<dependency>
```

```
<groupId>software.amazon.encryption.s3</groupId>
<artifactId>amazon-s3-encryption-client-java</artifactId>
<version>3.x</version>
</dependency>
```

<sup>2</sup> Il AWS Database Encryption Client per DynamoDB è disponibile utilizzando la seguente dipendenza Maven.

```
<dependency>
  <groupId>software.amazon.cryptography</groupId>
  <artifactId>aws-database-encryption-sdk-dynamodb</artifactId>
  <version>3.x</version>
</dependency>
```

## Dettagli sulla migrazione per librerie e utilità

- [S3 Transfer Manager](#)
- [Utilità per i metadati EC2](#)
- [CloudFrontpreassegnazione](#)
- [analisi URI S3](#)

## Modifiche al client

### costruttori di client

È necessario creare tutti i client utilizzando il metodo del generatore client. I costruttori non sono più disponibili.

### Example di creazione di un client nella versione 1.x

```
AmazonDynamoDB ddbClient = AmazonDynamoDBClientBuilder.defaultClient();
AmazonDynamoDBClient ddbClient = new AmazonDynamoDBClient();
```

### Example di creazione di un client nella versione 2.x

```
DynamoDbClient ddbClient = DynamoDbClient.create();
DynamoDbClient ddbClient = DynamoDbClient.builder().build();
```



## Nomi delle classi client

Tutti i nomi delle classi client ora contengono solo maiuscole e minuscole e non sono più preceduti da. Amazon Queste modifiche sono allineate ai nomi utilizzati nella AWS CLI.

### Example dei nomi delle classi in 1.x

```
AmazonDynamoDB
AWSACMPAAsyncClient
```

### Example dei nomi delle classi in 2.x

```
DynamoDbClient
AcmAsyncClient
```

### Modifiche al nome della classe client

Client 1.x	Client 2.x
<code>com.amazonaws.services.acmpca.AWSACMPAAsyncClient</code>	<code>software.amazon.awssdk.services.acm.AcmAsyncClient</code>
<code>com.amazonaws.services.acmpca.AWSACMPAClient</code>	<code>software.amazon.awssdk.services.acm.AcmClient</code>
<code>com.amazonaws.services.alexaforbusiness.AmazonAlexaForBusinessAsyncClient</code>	<code>software.amazon.awssdk.services.alexaforbusiness.AlexaForBusinessAsyncClient</code>
<code>com.amazonaws.services.alexaforbusiness.AmazonAlexaForBusinessClient</code>	<code>software.amazon.awssdk.services.alexaforbusiness.AlexaForBusinessClient</code>
<code>com.amazonaws.services.apigateway.AmazonApiGatewayAsyncClient</code>	<code>software.amazon.awssdk.services.apigateway.ApiGatewayAsyncClient</code>
<code>com.amazonaws.services.apigateway.AmazonApiGatewayClient</code>	<code>software.amazon.awssdk.services.apigateway.ApiGatewayClient</code>

Client 1.x	Client 2.x
<code>com.amazonaws.services.applicationautoscaling.AWSApplicationAutoScalingAsyncClient</code>	<code>software.amazon.awssdk.services.applicationautoscaling.ApplicationAutoScalingAsyncClient</code>
<code>com.amazonaws.services.applicationautoscaling.AWSApplicationAutoScalingClient</code>	<code>software.amazon.awssdk.services.applicationautoscaling.ApplicationAutoScalingClient</code>
<code>com.amazonaws.services.applicationdiscovery.AWSApplicationDiscoveryAsyncClient</code>	<code>software.amazon.awssdk.services.applicationdiscovery.ApplicationDiscoveryAsyncClient</code>
<code>com.amazonaws.services.applicationdiscovery.AWSApplicationDiscoveryClient</code>	<code>software.amazon.awssdk.services.applicationdiscovery.ApplicationDiscoveryClient</code>
<code>com.amazonaws.services.appstream.AmazonAppStreamAsyncClient</code>	<code>software.amazon.awssdk.services.appstream.AppStreamAsyncClient</code>
<code>com.amazonaws.services.appstream.AmazonAppStreamClient</code>	<code>software.amazon.awssdk.services.appstream.AppStreamClient</code>
<code>com.amazonaws.services.appsync.AWSAppSyncAsyncClient</code>	<code>software.amazon.awssdk.services.appsync.AppSyncAsyncClient</code>
<code>com.amazonaws.services.appsync.AWSAppSyncClient</code>	<code>software.amazon.awssdk.services.appsync.AppSyncClient</code>
<code>com.amazonaws.services.athena.AmazonAthenaAsyncClient</code>	<code>software.amazon.awssdk.services.athena.AthenaAsyncClient</code>
<code>com.amazonaws.services.athena.AmazonAthenaClient</code>	<code>software.amazon.awssdk.services.athena.AthenaClient</code>

Client 1.x	Client 2.x
<code>com.amazonaws.services.autoscaling.AmazonAutoScalingAsyncClient</code>	<code>software.amazon.awssdk.services.autoscaling.AutoScalingAsyncClient</code>
<code>com.amazonaws.services.autoscaling.AmazonAutoScalingClient</code>	<code>software.amazon.awssdk.services.autoscaling.AutoScalingClient</code>
<code>com.amazonaws.services.autoscalingplans.AWSAutoScalingPlansAsyncClient</code>	<code>software.amazon.awssdk.services.autoscalingplans.AutoScalingPlansAsyncClient</code>
<code>com.amazonaws.services.autoscalingplans.AWSAutoScalingPlansClient</code>	<code>software.amazon.awssdk.services.autoscalingplans.AutoScalingPlansClient</code>
<code>com.amazonaws.services.batch.AWSBatchAsyncClient</code>	<code>software.amazon.awssdk.services.batch.BatchAsyncClient</code>
<code>com.amazonaws.services.batch.AWSBatchClient</code>	<code>software.amazon.awssdk.services.batch.BatchClient</code>
<code>com.amazonaws.services.budgets.AWSBudgetsAsyncClient</code>	<code>software.amazon.awssdk.services.budgets.BudgetsAsyncClient</code>
<code>com.amazonaws.services.budgets.AWSBudgetsClient</code>	<code>software.amazon.awssdk.services.budgets.BudgetsClient</code>
<code>com.amazonaws.services.certificatemanager.AWSCertificateManagerAsyncClient</code>	<code>software.amazon.awssdk.services.acm.AcmAsyncClient</code>
<code>com.amazonaws.services.certificatemanager.AWSCertificateManagerClient</code>	<code>software.amazon.awssdk.services.acm.AcmClient</code>

Client 1.x	Client 2.x
<code>com.amazonaws.services.cloud9.AWSCloud9AsyncClient</code>	<code>software.amazon.awssdk.services.cloud9.Cloud9AsyncClient</code>
<code>com.amazonaws.services.cloud9.AWSCloud9Client</code>	<code>software.amazon.awssdk.services.cloud9.Cloud9Client</code>
<code>com.amazonaws.services.clouddirectory.AmazonCloudDirectoryAsyncClient</code>	<code>software.amazon.awssdk.services.clouddirectory.CloudDirectoryAsyncClient</code>
<code>com.amazonaws.services.clouddirectory.AmazonCloudDirectoryClient</code>	<code>software.amazon.awssdk.services.clouddirectory.CloudDirectoryClient</code>
<code>com.amazonaws.services.cloudformation.AmazonCloudFormationAsyncClient</code>	<code>software.amazon.awssdk.services.cloudformation.CloudFormationAsyncClient</code>
<code>com.amazonaws.services.cloudformation.AmazonCloudFormationClient</code>	<code>software.amazon.awssdk.services.cloudformation.CloudFormationClient</code>
<code>com.amazonaws.services.cloudfront.AmazonCloudFrontAsyncClient</code>	<code>software.amazon.awssdk.services.cloudfront.CloudFrontAsyncClient</code>
<code>com.amazonaws.services.cloudfront.AmazonCloudFrontClient</code>	<code>software.amazon.awssdk.services.cloudfront.CloudFrontClient</code>
<code>com.amazonaws.services.cloudhsm.AWSCloudHSMAsyncClient</code>	<code>software.amazon.awssdk.services.cloudhsm.CloudHsmAsyncClient</code>
<code>com.amazonaws.services.cloudhsm.AWSCloudHSMClient</code>	<code>software.amazon.awssdk.services.cloudhsm.CloudHsmClient</code>

Client 1.x	Client 2.x
<code>com.amazonaws.services.cloudhsmv2.AWSCloudHSMV2AsyncClient</code>	<code>software.amazon.awssdk.services.cloudhsmv2.CloudHsmV2AsyncClient</code>
<code>com.amazonaws.services.cloudhsmv2.AWSCloudHSMV2Client</code>	<code>software.amazon.awssdk.services.cloudhsmv2.CloudHsmV2Client</code>
<code>com.amazonaws.services.cloudsearchdomain.AmazonCloudSearchDomainAsyncClient</code>	<code>software.amazon.awssdk.services.cloudsearchdomain.CloudSearchDomainAsyncClient</code>
<code>com.amazonaws.services.cloudsearchdomain.AmazonCloudSearchDomainClient</code>	<code>software.amazon.awssdk.services.cloudsearchdomain.CloudSearchDomainClient</code>
<code>com.amazonaws.services.cloudsearchv2.AmazonCloudSearchAsyncClient</code>	<code>software.amazon.awssdk.services.cloudsearch.CloudSearchAsyncClient</code>
<code>com.amazonaws.services.cloudsearchv2.AmazonCloudSearchClient</code>	<code>software.amazon.awssdk.services.cloudsearch.CloudSearchClient</code>
<code>com.amazonaws.services.cloudtrail.AWSCloudTrailAsyncClient</code>	<code>software.amazon.awssdk.services.cloudtrail.CloudTrailAsyncClient</code>
<code>com.amazonaws.services.cloudtrail.AWSCloudTrailClient</code>	<code>software.amazon.awssdk.services.cloudtrail.CloudTrailClient</code>
<code>com.amazonaws.services.cloudwatch.AmazonCloudWatchAsyncClient</code>	<code>software.amazon.awssdk.services.cloudwatch.CloudWatchAsyncClient</code>

Client 1.x	Client 2.x
<code>com.amazonaws.services.cloudwatch.AmazonCloudWatchClient</code>	<code>software.amazon.awssdk.services.cloudwatch.CloudWatchClient</code>
<code>com.amazonaws.services.cloudwatchevents.AmazonCloudWatchEventsAsyncClient</code>	<code>software.amazon.awssdk.services.cloudwatchevents.CloudWatchEventsAsyncClient</code>
<code>com.amazonaws.services.cloudwatchevents.AmazonCloudWatchEventsClient</code>	<code>software.amazon.awssdk.services.cloudwatchevents.CloudWatchEventsClient</code>
<code>com.amazonaws.services.codebuild.AWSCodeBuildAsyncClient</code>	<code>software.amazon.awssdk.services.codebuild.CodeBuildAsyncClient</code>
<code>com.amazonaws.services.codebuild.AWSCodeBuildClient</code>	<code>software.amazon.awssdk.services.codebuild.CodeBuildClient</code>
<code>com.amazonaws.services.codecommit.AWSCodeCommitAsyncClient</code>	<code>software.amazon.awssdk.services.codecommit.CodeCommitAsyncClient</code>
<code>com.amazonaws.services.codecommit.AWSCodeCommitClient</code>	<code>software.amazon.awssdk.services.codecommit.CodeCommitClient</code>
<code>com.amazonaws.services.codedeploy.AmazonCodeDeployAsyncClient</code>	<code>software.amazon.awssdk.services.codedeploy.CodeDeployAsyncClient</code>
<code>com.amazonaws.services.codedeploy.AmazonCodeDeployClient</code>	<code>software.amazon.awssdk.services.codedeploy.CodeDeployClient</code>

Client 1.x	Client 2.x
<code>com.amazonaws.services.codepipeline.AWSCodePipelineAsyncClient</code>	<code>software.amazon.awssdk.services.codepipeline.CodePipelineAsyncClient</code>
<code>com.amazonaws.services.codepipeline.AWSCodePipelineClient</code>	<code>software.amazon.awssdk.services.codepipeline.CodePipelineClient</code>
<code>com.amazonaws.services.codestar.AWSCodeStarAsyncClient</code>	<code>software.amazon.awssdk.services.codestar.CodeStarAsyncClient</code>
<code>com.amazonaws.services.codestar.AWSCodeStarClient</code>	<code>software.amazon.awssdk.services.codestar.CodeStarClient</code>
<code>com.amazonaws.services.cognitoidentity.AmazonCognitoIdentityAsyncClient</code>	<code>software.amazon.awssdk.services.cognitoidentity.CognitoIdentityAsyncClient</code>
<code>com.amazonaws.services.cognitoidentity.AmazonCognitoIdentityClient</code>	<code>software.amazon.awssdk.services.cognitoidentity.CognitoIdentityClient</code>
<code>com.amazonaws.services.cognitoidentityprovider.AWSIdentityProviderAsyncClient</code>	<code>software.amazon.awssdk.services.cognitoidentityprovider.CognitoIdentityProviderAsyncClient</code>
<code>com.amazonaws.services.cognitoidentityprovider.AWSIdentityProviderClient</code>	<code>software.amazon.awssdk.services.cognitoidentityprovider.CognitoIdentityProviderClient</code>
<code>com.amazonaws.services.cognitosync.AmazonCognitoSyncAsyncClient</code>	<code>software.amazon.awssdk.services.cognitosync.CognitoSyncAsyncClient</code>

Client 1.x	Client 2.x
<code>com.amazonaws.services.cognitosync.AmazonCognitoSyncClient</code>	<code>software.amazon.awssdk.services.cognitosync.CognitoSyncClient</code>
<code>com.amazonaws.services.comprehend.AmazonComprehendAsyncClient</code>	<code>software.amazon.awssdk.services.comprehend.ComprehendAsyncClient</code>
<code>com.amazonaws.services.comprehend.AmazonComprehendClient</code>	<code>software.amazon.awssdk.services.comprehend.ComprehendClient</code>
<code>com.amazonaws.services.config.AmazonConfigAsyncClient</code>	<code>software.amazon.awssdk.services.config.ConfigAsyncClient</code>
<code>com.amazonaws.services.config.AmazonConfigClient</code>	<code>software.amazon.awssdk.services.config.ConfigClient</code>
<code>com.amazonaws.services.connect.AmazonConnectAsyncClient</code>	<code>software.amazon.awssdk.services.connect.ConnectAsyncClient</code>
<code>com.amazonaws.services.connect.AmazonConnectClient</code>	<code>software.amazon.awssdk.services.connect.ConnectClient</code>
<code>com.amazonaws.services.costandusagereport.AWSCostAndUsageReportAsyncClient</code>	<code>software.amazon.awssdk.services.costandusagereport.CostAndUsageReportAsyncClient</code>
<code>com.amazonaws.services.costandusagereport.AWSCostAndUsageReportClient</code>	<code>software.amazon.awssdk.services.costandusagereport.CostAndUsageReportClient</code>
<code>com.amazonaws.services.costexplorer.AWSCostExplorerAsyncClient</code>	<code>software.amazon.awssdk.services.costexplorer.CostExplorerAsyncClient</code>



Client 1.x	Client 2.x
<code>com.amazonaws.services.costexplorer.AWSCostExplorerClient</code>	<code>software.amazon.awssdk.services.costexplorer.CostExplorerClient</code>
<code>com.amazonaws.services.databasemigrationservice.AWSDatabaseMigrationServiceAsyncClient</code>	<code>software.amazon.awssdk.services.databasemigration.DatabaseMigrationAsyncClient</code>
<code>com.amazonaws.services.databasemigrationservice.AWSDatabaseMigrationServiceClient</code>	<code>software.amazon.awssdk.services.databasemigration.DatabaseMigrationClient</code>
<code>com.amazonaws.services.datapipeline.DataPipelineAsyncClient</code>	<code>software.amazon.awssdk.services.datapipeline.DataPipelineAsyncClient</code>
<code>com.amazonaws.services.datapipeline.DataPipelineClient</code>	<code>software.amazon.awssdk.services.datapipeline.DataPipelineAsyncClient</code>
<code>com.amazonaws.services.dax.AmazonDaxAsyncClient</code>	<code>software.amazon.awssdk.services.dax.DaxAsyncClient</code>
<code>com.amazonaws.services.dax.AmazonDaxClient</code>	<code>software.amazon.awssdk.services.dax.DaxClient</code>
<code>com.amazonaws.services.devicefarm.AWSDeviceFarmAsyncClient</code>	<code>software.amazon.awssdk.services.devicefarm.DeviceFarmAsyncClient</code>
<code>com.amazonaws.services.devicefarm.AWSDeviceFarmClient</code>	<code>software.amazon.awssdk.services.devicefarm.DeviceFarmClient</code>

Client 1.x	Client 2.x
<code>com.amazonaws.services.directconnect.AmazonDirectConnectAsyncClient</code>	<code>software.amazon.awssdk.services.directconnect.DirectConnectAsyncClient</code>
<code>com.amazonaws.services.directconnect.AmazonDirectConnectClient</code>	<code>software.amazon.awssdk.services.directconnect.DirectConnectClient</code>
<code>com.amazonaws.services.directory.AWSDirectoryServiceAsyncClient</code>	<code>software.amazon.awssdk.services.directory.DirectoryAsyncClient</code>
<code>com.amazonaws.services.directory.AWSDirectoryServiceClient</code>	<code>software.amazon.awssdk.services.directory.DirectoryClient</code>
<code>com.amazonaws.services.dlm.AmazonDLMAAsyncClient</code>	<code>software.amazon.awssdk.services.dlm.DlmAsyncClient</code>
<code>com.amazonaws.services.dlm.AmazonDLMClient</code>	<code>software.amazon.awssdk.services.dlm.DlmClient</code>
<code>com.amazonaws.services.dynamodbv2.AmazonDynamoDBAsyncClient</code>	<code>software.amazon.awssdk.services.dynamodb.DynamoDbAsyncClient</code>
<code>com.amazonaws.services.dynamodbv2.AmazonDynamoDBClient</code>	<code>software.amazon.awssdk.services.dynamodb.DynamoDbClient</code>
<code>com.amazonaws.services.dynamodbv2.AmazonDynamoDBStreamsAsyncClient</code>	<code>software.amazon.awssdk.services.dynamodb.streams.DynamoDbStreamsAsyncClient</code>
<code>com.amazonaws.services.dynamodbv2.AmazonDynamoDBStreamsClient</code>	<code>software.amazon.awssdk.services.dynamodb.streams.DynamoDbStreamsClient</code>

Client 1.x	Client 2.x
<code>com.amazonaws.services.ec2. AmazonEC2AsyncClient</code>	<code>software.amazon.awssdk.serv ices.ec2.Ec2AsyncClient</code>
<code>com.amazonaws.services.ec2. AmazonEC2Client</code>	<code>software.amazon.awssdk.serv ices.ec2.Ec2Client</code>
<code>com.amazonaws.services.ecr. AmazonECRAsyncClient</code>	<code>software.amazon.awssdk.serv ices.ecr.EcrAsyncClient</code>
<code>com.amazonaws.services.ecr. AmazonECRClient</code>	<code>software.amazon.awssdk.serv ices.ecr.EcrClient</code>
<code>com.amazonaws.services.ecs. AmazonECSAsyncClient</code>	<code>software.amazon.awssdk.serv ices.ecs.EcsAsyncClient</code>
<code>com.amazonaws.services.ecs. AmazonECSClient</code>	<code>software.amazon.awssdk.serv ices.ecs.EcsClient</code>
<code>com.amazonaws.services.eks. AmazonEKSAsyncClient</code>	<code>software.amazon.awssdk.serv ices.eks.EksAsyncClient</code>
<code>com.amazonaws.services.eks. AmazonEKSClient</code>	<code>software.amazon.awssdk.serv ices.eks.EksClient</code>
<code>com.amazonaws.services.elas ticache.AmazonElasticCacheAs yncClient</code>	<code>software.amazon.awssdk.serv ices.elasticache.ElastiCach eAsyncClient</code>
<code>com.amazonaws.services.elas ticache.AmazonElasticCacheClient</code>	<code>software.amazon.awssdk.serv ices.elasticache.ElastiCach eClient</code>
<code>com.amazonaws.services.elas ticbeanstalk.AWSElasticBean stalkAsyncClient</code>	<code>software.amazon.awssdk.serv ices.elasticbeanstalk.Elast icBeanstalkAsyncClient</code>

Client 1.x	Client 2.x
<code>com.amazonaws.services.elasticbeanstalk.AWSElasticBeanstalkClient</code>	<code>software.amazon.awssdk.services.elasticbeanstalk.ElasticBeanstalkClient</code>
<code>com.amazonaws.services.elasticfilesystem.AmazonElasticFileSystemAsyncClient</code>	<code>software.amazon.awssdk.services.efs.EfsAsyncClient</code>
<code>com.amazonaws.services.elasticfilesystem.AmazonElasticFileSystemClient</code>	<code>software.amazon.awssdk.services.efs.EfsClient</code>
<code>com.amazonaws.services.elasticloadbalancing.AmazonElasticLoadBalancingAsyncClient</code>	<code>software.amazon.awssdk.services.elasticloadbalancing.ElasticLoadBalancingAsyncClient</code>
<code>com.amazonaws.services.elasticloadbalancing.AmazonElasticLoadBalancingClient</code>	<code>software.amazon.awssdk.services.elasticloadbalancing.ElasticLoadBalancingClient</code>
<code>com.amazonaws.services.elasticloadbalancingv2.AmazonElasticLoadBalancingV2AsyncClient</code>	<code>software.amazon.awssdk.services.elasticloadbalancingv2.ElasticLoadBalancingV2AsyncClient</code>
<code>com.amazonaws.services.elasticloadbalancingv2.AmazonElasticLoadBalancingV2Client</code>	<code>software.amazon.awssdk.services.elasticloadbalancingv2.ElasticLoadBalancingV2Client</code>
<code>com.amazonaws.services.elasticmapreduce.AmazonElasticMapReduceAsyncClient</code>	<code>software.amazon.awssdk.services.emr.EmrAsyncClient</code>
<code>com.amazonaws.services.elasticmapreduce.AmazonElasticMapReduceClient</code>	<code>software.amazon.awssdk.services.emr.EmrClient</code>

Client 1.x	Client 2.x
<code>com.amazonaws.services.elasticsearch.AWSElasticsearchAsyncClient</code>	<code>software.amazon.awssdk.services.elasticsearch.ElasticsearchAsyncClient</code>
<code>com.amazonaws.services.elasticsearch.AWSElasticsearchClient</code>	<code>software.amazon.awssdk.services.elasticsearch.ElasticsearchClient</code>
<code>com.amazonaws.services.elastictranscoder.AmazonElasticTranscoderAsyncClient</code>	<code>software.amazon.awssdk.services.elastictranscoder.ElasticTranscoderAsyncClient</code>
<code>com.amazonaws.services.elastictranscoder.AmazonElasticTranscoderClient</code>	<code>software.amazon.awssdk.services.elastictranscoder.ElasticTranscoderClient</code>
<code>com.amazonaws.services.fms.AWSFMSAsyncClient</code>	<code>software.amazon.awssdk.services.fms.FmsAsyncClient</code>
<code>com.amazonaws.services.fms.AWSFMSClient</code>	<code>software.amazon.awssdk.services.fms.FmsClient</code>
<code>com.amazonaws.services.gamelift.AmazonGameLiftAsyncClient</code>	<code>software.amazon.awssdk.services.gamelift.GameLiftAsyncClient</code>
<code>com.amazonaws.services.gamelift.AmazonGameLiftClient</code>	<code>software.amazon.awssdk.services.gamelift.GameLiftClient</code>
<code>com.amazonaws.services.glacier.AmazonGlacierAsyncClient</code>	<code>software.amazon.awssdk.services.glacier.GlacierAsyncClient</code>
<code>com.amazonaws.services.glacier.AmazonGlacierClient</code>	<code>software.amazon.awssdk.services.glacier.GlacierClient</code>
<code>com.amazonaws.services.glue.AWSGlueAsyncClient</code>	<code>software.amazon.awssdk.services.glue.GlueAsyncClient</code>

Client 1.x	Client 2.x
<code>com.amazonaws.services.glue.AWSGlueClient</code>	<code>software.amazon.awssdk.services.glue.GlueClient</code>
<code>com.amazonaws.services.greengrass.AWSGreengrassAsyncClient</code>	<code>software.amazon.awssdk.services.greengrass.GreengrassAsyncClient</code>
<code>com.amazonaws.services.greengrass.AWSGreengrassClient</code>	<code>software.amazon.awssdk.services.greengrass.GreengrassClient</code>
<code>com.amazonaws.services.guardduty.AmazonGuardDutyAsyncClient</code>	<code>software.amazon.awssdk.services.guardduty.GuardDutyAsyncClient</code>
<code>com.amazonaws.services.guardduty.AmazonGuardDutyClient</code>	<code>software.amazon.awssdk.services.guardduty.GuardDutyClient</code>
<code>com.amazonaws.services.health.AWSHealthAsyncClient</code>	<code>software.amazon.awssdk.services.health.HealthAsyncClient</code>
<code>com.amazonaws.services.health.AWSHealthClient</code>	<code>software.amazon.awssdk.services.health.HealthClient</code>
<code>com.amazonaws.services.identitymanagement.AmazonIdentityManagementAsyncClient</code>	<code>software.amazon.awssdk.services.iam.IamAsyncClient</code>
<code>com.amazonaws.services.identitymanagement.AmazonIdentityManagementClient</code>	<code>software.amazon.awssdk.services.iam.IamClient</code>
<code>com.amazonaws.services.importexport.AmazonImportExportAsyncClient</code>	<code>software.amazon.awssdk.services.importexport.ImportExportAsyncClient</code>

Client 1.x	Client 2.x
<code>com.amazonaws.services.importexport.AmazonImportExportClient</code>	<code>software.amazon.awssdk.services.importexport.ImportExportClient</code>
<code>com.amazonaws.services.inspector.AmazonInspectorAsyncClient</code>	<code>software.amazon.awssdk.services.inspector.InspectorAsyncClient</code>
<code>com.amazonaws.services.inspector.AmazonInspectorClient</code>	<code>software.amazon.awssdk.services.inspector.InspectorClient</code>
<code>com.amazonaws.services.iot.AWSIoTAsyncClient</code>	<code>software.amazon.awssdk.services.iot.IotAsyncClient</code>
<code>com.amazonaws.services.iot.AWSIoTClient</code>	<code>software.amazon.awssdk.services.iot.IotClient</code>
<code>com.amazonaws.services.iot1clickdevices.AWSIoT1ClickDevicesAsyncClient</code>	<code>software.amazon.awssdk.services.iot1clickdevices.Iot1ClickDevicesAsyncClient</code>
<code>com.amazonaws.services.iot1clickdevices.AWSIoT1ClickDevicesClient</code>	<code>software.amazon.awssdk.services.iot1clickdevices.Iot1ClickDevicesClient</code>
<code>com.amazonaws.services.iot1clickprojects.AWSIoT1ClickProjectsAsyncClient</code>	<code>software.amazon.awssdk.services.iot1clickprojects.Iot1ClickProjectsAsyncClient</code>
<code>com.amazonaws.services.iot1clickprojects.AWSIoT1ClickProjectsClient</code>	<code>software.amazon.awssdk.services.iot1clickprojects.Iot1ClickProjectsClient</code>
<code>com.amazonaws.services.iotanalytics.AWSIoTAnalyticsAsyncClient</code>	<code>software.amazon.awssdk.services.iotanalytics.IotAnalyticsAsyncClient</code>

Client 1.x	Client 2.x
<code>com.amazonaws.services.iotanalytics.AWSIoTAnalyticsClient</code>	<code>software.amazon.awssdk.services.iotanalytics.IotAnalyticsClient</code>
<code>com.amazonaws.services.iotdata.AWSIoTDataAsyncClient</code>	<code>software.amazon.awssdk.services.iotdata.IotDataAsyncClient</code>
<code>com.amazonaws.services.iotdata.AWSIoTDataClient</code>	<code>software.amazon.awssdk.services.iotdata.IotDataClient</code>
<code>com.amazonaws.services.iotjobsdataplane.AWSIoTJobsDataPlaneAsyncClient</code>	<code>software.amazon.awssdk.services.iotdataplane.IotDataPlaneAsyncClient</code>
<code>com.amazonaws.services.iotjobsdataplane.AWSIoTJobsDataPlaneClient</code>	<code>software.amazon.awssdk.services.iotdataplane.IotDataPlaneClient</code>
<code>com.amazonaws.services.kinesis.AmazonKinesisAsyncClient</code>	<code>software.amazon.awssdk.services.kinesis.KinesisAsyncClient</code>
<code>com.amazonaws.services.kinesis.AmazonKinesisClient</code>	<code>software.amazon.awssdk.services.kinesis.KinesisClient</code>
<code>com.amazonaws.services.kinesisanalytics.AmazonKinesisAnalyticsAsyncClient</code>	<code>software.amazon.awssdk.services.kinesisanalytics.KinesisAnalyticsAsyncClient</code>
<code>com.amazonaws.services.kinesisanalytics.AmazonKinesisAnalyticsClient</code>	<code>software.amazon.awssdk.services.kinesisanalytics.KinesisAnalyticsClient</code>
<code>com.amazonaws.services.kinesisfirehose.AmazonKinesisFirehoseAsyncClient</code>	<code>software.amazon.awssdk.services.firehose.FirehoseAsyncClient</code>



Client 1.x	Client 2.x
<code>com.amazonaws.services.kinesisfirehose.AmazonKinesisFirehoseClient</code>	<code>software.amazon.awssdk.services.firehose.FirehoseClient</code>
<code>com.amazonaws.services.kinesisvideo.AmazonKinesisVideoArchivedMediaAsyncClient</code>	<code>software.amazon.awssdk.services.kinesisvideoarchivedmedia.KinesisVideoArchivedMediaAsyncClient</code>
<code>com.amazonaws.services.kinesisvideo.AmazonKinesisVideoArchivedMediaClient</code>	<code>software.amazon.awssdk.services.kinesisvideoarchivedmedia.KinesisVideoArchivedMediaClient</code>
<code>com.amazonaws.services.kinesisvideo.AmazonKinesisVideoAsyncClient</code>	<code>software.amazon.awssdk.services.kinesisvideo.KinesisVideoAsyncClient</code>
<code>com.amazonaws.services.kinesisvideo.AmazonKinesisVideoClient</code>	<code>software.amazon.awssdk.services.kinesisvideo.KinesisVideoClient</code>
<code>com.amazonaws.services.kinesisvideo.AmazonKinesisVideoMediaAsyncClient</code>	<code>software.amazon.awssdk.services.kinesisvideomedia.KinesisVideoMediaAsyncClient</code>
<code>com.amazonaws.services.kinesisvideo.AmazonKinesisVideoMediaClient</code>	<code>software.amazon.awssdk.services.kinesisvideomedia.KinesisVideoMediaClient</code>
<code>com.amazonaws.services.kinesisvideo.AmazonKinesisVideoPutMediaClient</code>	Non supportato
<code>com.amazonaws.services.kms.AWSKMSAsyncClient</code>	<code>software.amazon.awssdk.services.kms.KmsAsyncClient</code>

Client 1.x	Client 2.x
<code>com.amazonaws.services.kms.AWSKMSClient</code>	<code>software.amazon.awssdk.services.kms.KmsClient</code>
<code>com.amazonaws.services.lambda.AWSLambdaAsyncClient</code>	<code>software.amazon.awssdk.services.lambda.LambdaAsyncClient</code>
<code>com.amazonaws.services.lambda.AWSLambdaClient</code>	<code>software.amazon.awssdk.services.lambda.LambdaClient</code>
<code>com.amazonaws.services.lexmodelbuilding.AmazonLexModelBuildingAsyncClient</code>	<code>software.amazon.awssdk.services.lexmodelbuilding.LexModelBuildingAsyncClient</code>
<code>com.amazonaws.services.lexmodelbuilding.AmazonLexModelBuildingClient</code>	<code>software.amazon.awssdk.services.lexmodelbuilding.LexModelBuildingClient</code>
<code>com.amazonaws.services.lexruntime.AmazonLexRuntimeAsyncClient</code>	<code>software.amazon.awssdk.services.lexruntime.LexRuntimeAsyncClient</code>
<code>com.amazonaws.services.lexruntime.AmazonLexRuntimeClient</code>	<code>software.amazon.awssdk.services.lexruntime.LexRuntimeClient</code>
<code>com.amazonaws.services.lightsail.AmazonLightsailAsyncClient</code>	<code>software.amazon.awssdk.services.lightsail.LightsailAsyncClient</code>
<code>com.amazonaws.services.lightsail.AmazonLightsailClient</code>	<code>software.amazon.awssdk.services.lightsail.LightsailClient</code>
<code>com.amazonaws.services.logs.AWSLogsAsyncClient</code>	<code>software.amazon.awssdk.services.logs.LogsAsyncClient</code>
<code>com.amazonaws.services.logs.AWSLogsClient</code>	<code>software.amazon.awssdk.services.logs.LogsClient</code>

Client 1.x	Client 2.x
<code>com.amazonaws.services.machinelearning.AmazonMachineLearningAsyncClient</code>	<code>software.amazon.awssdk.services.machinelearning.MachineLearningAsyncClient</code>
<code>com.amazonaws.services.machinelearning.AmazonMachineLearningClient</code>	<code>software.amazon.awssdk.services.machinelearning.MachineLearningClient</code>
<code>com.amazonaws.services.macie.AmazonMacieAsyncClient</code>	<code>software.amazon.awssdk.services.macie.MacieAsyncClient</code>
<code>com.amazonaws.services.macie.AmazonMacieClient</code>	<code>software.amazon.awssdk.services.macie.MacieClient</code>
<code>com.amazonaws.services.marketplacecommerceanalytics.AWSMarketplaceCommerceAnalyticsAsyncClient</code>	<code>software.amazon.awssdk.services.marketplacecommerceanalytics.MarketplaceCommerceAnalyticsAsyncClient</code>
<code>com.amazonaws.services.marketplacecommerceanalytics.AWSMarketplaceCommerceAnalyticsClient</code>	<code>software.amazon.awssdk.services.marketplacecommerceanalytics.MarketplaceCommerceAnalyticsClient</code>
<code>com.amazonaws.services.marketplaceentitlement.AWSMarketplaceEntitlementAsyncClient</code>	<code>software.amazon.awssdk.services.marketplaceentitlement.MarketplaceEntitlementAsyncClient</code>
<code>com.amazonaws.services.marketplaceentitlement.AWSMarketplaceEntitlementClient</code>	<code>software.amazon.awssdk.services.marketplaceentitlement.MarketplaceEntitlementClient</code>
<code>com.amazonaws.services.marketplacemetering.AWSMarketplaceMeteringAsyncClient</code>	<code>software.amazon.awssdk.services.marketplacemetering.MarketplaceMeteringAsyncClient</code>

Client 1.x	Client 2.x
<code>com.amazonaws.services.marketplacemetering.AWSMarketplaceMeteringClient</code>	<code>software.amazon.awssdk.services.marketplacemetering.MarketplaceMeteringClient</code>
<code>com.amazonaws.services.mediaconvert.AWSMediaConvertAsyncClient</code>	<code>software.amazon.awssdk.services.mediaconvert.MediaConvertAsyncClient</code>
<code>com.amazonaws.services.mediaconvert.AWSMediaConvertClient</code>	<code>software.amazon.awssdk.services.mediaconvert.MediaConvertClient</code>
<code>com.amazonaws.services.medialive.AWSMediaLiveAsyncClient</code>	<code>software.amazon.awssdk.services.medialive.MediaLiveAsyncClient</code>
<code>com.amazonaws.services.medialive.AWSMediaLiveClient</code>	<code>software.amazon.awssdk.services.medialive.MediaLiveClient</code>
<code>com.amazonaws.services.mediapackage.AWSMediaPackageAsyncClient</code>	<code>software.amazon.awssdk.services.mediapackage.MediaPackageAsyncClient</code>
<code>com.amazonaws.services.mediapackage.AWSMediaPackageClient</code>	<code>software.amazon.awssdk.services.mediapackage.MediaPackageClient</code>
<code>com.amazonaws.services.mediastore.AWSMediaStoreAsyncClient</code>	<code>software.amazon.awssdk.services.mediastore.MediaStoreAsyncClient</code>
<code>com.amazonaws.services.mediastore.AWSMediaStoreClient</code>	<code>software.amazon.awssdk.services.mediastore.MediaStoreClient</code>

Client 1.x	Client 2.x
<code>com.amazonaws.services.mediastoredata.AWSMediaStoreDataAsyncClient</code>	<code>software.amazon.awssdk.services.mediastoredata.MediaStoreDataAsyncClient</code>
<code>com.amazonaws.services.mediastoredata.AWSMediaStoreDataClient</code>	<code>software.amazon.awssdk.services.mediastoredata.MediaStoreDataClient</code>
<code>com.amazonaws.services.mediatailor.AWSMediaTailorAsyncClient</code>	<code>software.amazon.awssdk.services.mediatailor.MediaTailorAsyncClient</code>
<code>com.amazonaws.services.mediatailor.AWSMediaTailorClient</code>	<code>software.amazon.awssdk.services.mediatailor.MediaTailorClient</code>
<code>com.amazonaws.services.migrationhub.AWSMigrationHubAsyncClient</code>	<code>software.amazon.awssdk.services.migrationhub.MigrationHubAsyncClient</code>
<code>com.amazonaws.services.migrationhub.AWSMigrationHubClient</code>	<code>software.amazon.awssdk.services.migrationhub.MigrationHubClient</code>
<code>com.amazonaws.services.mobile.AWSMobileAsyncClient</code>	<code>software.amazon.awssdk.services.mobile.MobileAsyncClient</code>
<code>com.amazonaws.services.mobile.AWSMobileClient</code>	<code>software.amazon.awssdk.services.mobile.MobileClient</code>
<code>com.amazonaws.services.mq.AmazonMQAsyncClient</code>	<code>software.amazon.awssdk.services.mq.MqAsyncClient</code>
<code>com.amazonaws.services.mq.AmazonMQClient</code>	<code>software.amazon.awssdk.services.mq.MqClient</code>

Client 1.x	Client 2.x
<code>com.amazonaws.services.mturk.AmazonMTurkAsyncClient</code>	<code>software.amazon.awssdk.services.mturk.MTurkAsyncClient</code>
<code>com.amazonaws.services.mturk.AmazonMTurkClient</code>	<code>software.amazon.awssdk.services.mturk.MTurkClient</code>
<code>com.amazonaws.services.neptune.AmazonNeptuneAsyncClient</code>	<code>software.amazon.awssdk.services.neptune.NeptuneAsyncClient</code>
<code>com.amazonaws.services.neptune.AmazonNeptuneClient</code>	<code>software.amazon.awssdk.services.neptune.NeptuneClient</code>
<code>com.amazonaws.services.opsworks.AWSOpsWorksAsyncClient</code>	<code>software.amazon.awssdk.services.opsworks.OpsWorksAsyncClient</code>
<code>com.amazonaws.services.opsworks.AWSOpsWorksClient</code>	<code>software.amazon.awssdk.services.opsworks.OpsWorksClient</code>
<code>com.amazonaws.services.opsworkscm.AWSOpsWorksCMAsyncClient</code>	<code>software.amazon.awssdk.services.opsworkscm.OpsWorksCmAsyncClient</code>
<code>com.amazonaws.services.opsworkscm.AWSOpsWorksCMClient</code>	<code>software.amazon.awssdk.services.opsworkscm.OpsWorksCmClient</code>
<code>com.amazonaws.services.organizations.AWSOrganizationsAsyncClient</code>	<code>software.amazon.awssdk.services.organizations.OrganizationsAsyncClient</code>
<code>com.amazonaws.services.organizations.AWSOrganizationsClient</code>	<code>software.amazon.awssdk.services.organizations.OrganizationsClient</code>
<code>com.amazonaws.services.pi.AWSPIAsyncClient</code>	<code>software.amazon.awssdk.services.pi.PiAsyncClient</code>

Client 1.x	Client 2.x
<code>com.amazonaws.services.pi.AWSPIClient</code>	<code>software.amazon.awssdk.services.pi.PiClient</code>
<code>com.amazonaws.services.pinpoint.AmazonPinpointAsyncClient</code>	<code>software.amazon.awssdk.services.pinpoint.PinpointAsyncClient</code>
<code>com.amazonaws.services.pinpoint.AmazonPinpointClient</code>	<code>software.amazon.awssdk.services.pinpoint.PinpointClient</code>
<code>com.amazonaws.services.polly.AmazonPollyAsyncClient</code>	<code>software.amazon.awssdk.services.polly.PollyAsyncClient</code>
<code>com.amazonaws.services.polly.AmazonPollyClient</code>	<code>software.amazon.awssdk.services.polly.PollyClient</code>
<code>com.amazonaws.services.pricing.AWS PricingAsyncClient</code>	<code>software.amazon.awssdk.services.pricing.PricingAsyncClient</code>
<code>com.amazonaws.services.pricing.AWS PricingClient</code>	<code>software.amazon.awssdk.services.pricing.PricingClient</code>
<code>com.amazonaws.services.rds.AmazonRDSAsyncClient</code>	<code>software.amazon.awssdk.services.rds.RdsAsyncClient</code>
<code>com.amazonaws.services.rds.AmazonRDSClient</code>	<code>software.amazon.awssdk.services.rds.RdsClient</code>
<code>com.amazonaws.services.redshift.AmazonRedshiftAsyncClient</code>	<code>software.amazon.awssdk.services.redshift.RedshiftAsyncClient</code>
<code>com.amazonaws.services.redshift.AmazonRedshiftClient</code>	<code>software.amazon.awssdk.services.redshift.RedshiftClient</code>

Client 1.x	Client 2.x
<code>com.amazonaws.services.rekognition.AmazonRekognitionAsyncClient</code>	<code>software.amazon.awssdk.services.rekognition.RekognitionAsyncClient</code>
<code>com.amazonaws.services.rekognition.AmazonRekognitionClient</code>	<code>software.amazon.awssdk.services.rekognition.RekognitionClient</code>
<code>com.amazonaws.services.resourcegroups.AWSResourceGroupsAsyncClient</code>	<code>software.amazon.awssdk.services.resourcegroups.ResourceGroupsAsyncClient</code>
<code>com.amazonaws.services.resourcegroups.AWSResourceGroupsClient</code>	<code>software.amazon.awssdk.services.resourcegroups.ResourceGroupsClient</code>
<code>com.amazonaws.services.resourcegroupstaggingapi.AWSResourceGroupsTaggingAPIAsyncClient</code>	<code>software.amazon.awssdk.services.resourcegroupstaggingapi.ResourceGroupsTaggingApiAsyncClient</code>
<code>com.amazonaws.services.resourcegroupstaggingapi.AWSResourceGroupsTaggingAPIClient</code>	<code>software.amazon.awssdk.services.resourcegroupstaggingapi.ResourceGroupsTaggingApiClient</code>
<code>com.amazonaws.services.route53.AmazonRoute53AsyncClient</code>	<code>software.amazon.awssdk.services.route53.Route53AsyncClient</code>
<code>com.amazonaws.services.route53.AmazonRoute53Client</code>	<code>software.amazon.awssdk.services.route53.Route53Client</code>
<code>com.amazonaws.services.route53domains.AmazonRoute53DomainsAsyncClient</code>	<code>software.amazon.awssdk.services.route53domains.Route53DomainsAsyncClient</code>



Client 1.x	Client 2.x
<code>com.amazonaws.services.route53domains.AmazonRoute53DomainsClient</code>	<code>software.amazon.awssdk.services.route53domains.Route53DomainsClient</code>
<code>com.amazonaws.services.s3.AmazonS3Client</code>	<code>software.amazon.awssdk.services.s3.S3Client</code>
<code>com.amazonaws.services.sagemaker.AmazonSageMakerAsyncClient</code>	<code>software.amazon.awssdk.services.sagemaker.SageMakerAsyncClient</code>
<code>com.amazonaws.services.sagemaker.AmazonSageMakerClient</code>	<code>software.amazon.awssdk.services.sagemaker.SageMakerClient</code>
<code>com.amazonaws.services.sagemakerruntime.AmazonSageMakerRuntimeAsyncClient</code>	<code>software.amazon.awssdk.services.sagemakerruntime.SageMakerRuntimeAsyncClient</code>
<code>com.amazonaws.services.sagemakerruntime.AmazonSageMakerRuntimeClient</code>	<code>software.amazon.awssdk.services.sagemakerruntime.SageMakerRuntimeClient</code>
<code>com.amazonaws.services.secretsmanager.AWSSecretsManagerAsyncClient</code>	<code>software.amazon.awssdk.services.secretsmanager.SecretsManagerAsyncClient</code>
<code>com.amazonaws.services.secretsmanager.AWSSecretsManagerClient</code>	<code>software.amazon.awssdk.services.secretsmanager.SecretsManagerClient</code>
<code>com.amazonaws.services.securitytoken.AWSSecurityTokenServiceAsyncClient</code>	<code>software.amazon.awssdk.services.sts.StsAsyncClient</code>
<code>com.amazonaws.services.securitytoken.AWSSecurityTokenServiceClient</code>	<code>software.amazon.awssdk.services.sts.StsClient</code>

Client 1.x	Client 2.x
<code>com.amazonaws.services.serverlessapplicationrepository.AWSServerlessApplicationRepositoryAsyncClient</code>	<code>software.amazon.awssdk.services.serverlessapplicationrepository.ServerlessApplicationRepositoryAsyncClient</code>
<code>com.amazonaws.services.serverlessapplicationrepository.AWSServerlessApplicationRepositoryClient</code>	<code>software.amazon.awssdk.services.serverlessapplicationrepository.ServerlessApplicationRepositoryClient</code>
<code>com.amazonaws.services.servermigration.AWSServerMigrationAsyncClient</code>	<code>software.amazon.awssdk.services.sms.SmsAsyncClient</code>
<code>com.amazonaws.services.servermigration.AWSServerMigrationClient</code>	<code>software.amazon.awssdk.services.sms.SmsClient</code>
<code>com.amazonaws.services.servicecatalog.AWSServiceCatalogAsyncClient</code>	<code>software.amazon.awssdk.services.servicecatalog.ServiceCatalogAsyncClient</code>
<code>com.amazonaws.services.servicecatalog.AWSServiceCatalogClient</code>	<code>software.amazon.awssdk.services.servicecatalog.ServiceCatalogClient</code>
<code>com.amazonaws.services.servicediscovery.AWSServiceDiscoveryAsyncClient</code>	<code>software.amazon.awssdk.services.servicediscovery.ServiceDiscoveryAsyncClient</code>
<code>com.amazonaws.services.servicediscovery.AWSServiceDiscoveryClient</code>	<code>software.amazon.awssdk.services.servicediscovery.ServiceDiscoveryClient</code>
<code>com.amazonaws.services.shield.AWSShieldAsyncClient</code>	<code>software.amazon.awssdk.services.shield.ShieldAsyncClient</code>

Client 1.x	Client 2.x
<code>com.amazonaws.services.shield.AWSShieldClient</code>	<code>software.amazon.awssdk.services.shield.ShieldClient</code>
<code>com.amazonaws.services.simpledb.AmazonSimpleDBAsyncClient</code>	<code>software.amazon.awssdk.services.simpledb.SimpleDbAsyncClient</code>
<code>com.amazonaws.services.simpledb.AmazonSimpleDBClient</code>	<code>software.amazon.awssdk.services.simpledb.SimpleDbClient</code>
<code>com.amazonaws.services.simpleemail.AmazonSimpleEmailServiceAsyncClient</code>	<code>software.amazon.awssdk.services.ses.SesAsyncClient</code>
<code>com.amazonaws.services.simpleemail.AmazonSimpleEmailServiceClient</code>	<code>software.amazon.awssdk.services.ses.SesClient</code>
<code>com.amazonaws.services.simplesystemsmanagement.AWSSimpleSystemsManagementAsyncClient</code>	<code>software.amazon.awssdk.services.ssm.SsmAsyncClient</code>
<code>com.amazonaws.services.simplesystemsmanagement.AWSSimpleSystemsManagementClient</code>	<code>software.amazon.awssdk.services.ssm.SsmClient</code>
<code>com.amazonaws.services.simpleworkflow.AmazonSimpleWorkflowAsyncClient</code>	<code>software.amazon.awssdk.services.swf.SwfAsyncClient</code>
<code>com.amazonaws.services.simpleworkflow.AmazonSimpleWorkflowClient</code>	<code>software.amazon.awssdk.services.swf.SwfClient</code>
<code>com.amazonaws.services.snowball.AmazonSnowballAsyncClient</code>	<code>software.amazon.awssdk.services.snowball.SnowballAsyncClient</code>

Client 1.x	Client 2.x
<code>com.amazonaws.services.snowball.AmazonSnowballClient</code>	<code>software.amazon.awssdk.services.snowball.SnowballClient</code>
<code>com.amazonaws.services.sns.AmazonSNSAsyncClient</code>	<code>software.amazon.awssdk.services.sns.SnsAsyncClient</code>
<code>com.amazonaws.services.sns.AmazonSNSClient</code>	<code>software.amazon.awssdk.services.sns.SnsClient</code>
<code>com.amazonaws.services.sqs.AmazonSQSAsyncClient</code>	<code>software.amazon.awssdk.services.sqs.SqsAsyncClient</code>
<code>com.amazonaws.services.sqs.AmazonSQSClient</code>	<code>software.amazon.awssdk.services.sqs.SqsClient</code>
<code>com.amazonaws.services.stepfunctions.AWSStepFunctionsAsyncClient</code>	<code>software.amazon.awssdk.services.sfn.SfnAsyncClient</code>
<code>com.amazonaws.services.stepfunctions.AWSStepFunctionsClient</code>	<code>software.amazon.awssdk.services.sfn.SfnClient</code>
<code>com.amazonaws.services.storagegateway.AWSStorageGatewayAsyncClient</code>	<code>software.amazon.awssdk.services.storagegateway.StorageGatewayAsyncClient</code>
<code>com.amazonaws.services.storagegateway.AWSStorageGatewayClient</code>	<code>software.amazon.awssdk.services.storagegateway.StorageGatewayClient</code>
<code>com.amazonaws.services.support.AWSSupportAsyncClient</code>	<code>software.amazon.awssdk.services.support.SupportAsyncClient</code>
<code>com.amazonaws.services.support.AWSSupportClient</code>	<code>software.amazon.awssdk.services.support.SupportClient</code>

Client 1.x	Client 2.x
<code>com.amazonaws.services.transcribe.AmazonTranscribeAsyncClient</code>	<code>software.amazon.awssdk.services.transcribe.TranscribeAsyncClient</code>
<code>com.amazonaws.services.transcribe.AmazonTranscribeClient</code>	<code>software.amazon.awssdk.services.transcribe.TranscribeClient</code>
<code>com.amazonaws.services.translate.AmazonTranslateAsyncClient</code>	<code>software.amazon.awssdk.services.translate.TranslateAsyncClient</code>
<code>com.amazonaws.services.translate.AmazonTranslateClient</code>	<code>software.amazon.awssdk.services.translate.TranslateClient</code>
<code>com.amazonaws.services.waf.AWSWAFAsyncClient</code>	<code>software.amazon.awssdk.services.waf.WafAsyncClient</code>
<code>com.amazonaws.services.waf.AWSWAFClient</code>	<code>software.amazon.awssdk.services.waf.WafClient</code>
<code>com.amazonaws.services.waf.AWSWAFRegionalAsyncClient</code>	<code>software.amazon.awssdk.services.waf.regional.WafRegionalAsyncClient</code>
<code>com.amazonaws.services.waf.AWSWAFRegionalClient</code>	<code>software.amazon.awssdk.services.waf.regional.WafRegionalClient</code>
<code>com.amazonaws.services.workdocs.AmazonWorkDocsAsyncClient</code>	<code>software.amazon.awssdk.services.workdocs.WorkDocsAsyncClient</code>
<code>com.amazonaws.services.workdocs.AmazonWorkDocsClient</code>	<code>software.amazon.awssdk.services.workdocs.WorkDocsClient</code>

Client 1.x	Client 2.x
<code>com.amazonaws.services.workmail.AmazonWorkMailAsyncClient</code>	<code>software.amazon.awssdk.services.workmail.WorkMailAsyncClient</code>
<code>com.amazonaws.services.workmail.AmazonWorkMailClient</code>	<code>software.amazon.awssdk.services.workmail.WorkMailClient</code>
<code>com.amazonaws.services.workspaces.AmazonWorkspacesAsyncClient</code>	<code>software.amazon.awssdk.services.workspaces.WorkSpacesAsyncClient</code>
<code>com.amazonaws.services.workspaces.AmazonWorkspacesClient</code>	<code>software.amazon.awssdk.services.workspaces.WorkSpacesClient</code>
<code>com.amazonaws.services.xray.AWSXRayAsyncClient</code>	<code>software.amazon.awssdk.services.xray.XRayAsyncClient</code>
<code>com.amazonaws.services.xray.AWSXRayClient</code>	<code>software.amazon.awssdk.services.xray.XRayClient</code>

## Impostazioni predefinite per la creazione del client

Nella versione 2.x, sono state apportate le seguenti modifiche alla logica di creazione del client predefinita.

- La catena di provider di credenziali predefinita per S3 non include più credenziali anonime. È necessario specificare manualmente l'accesso anonimo a S3 utilizzando `AnonymousCredentialsProvider`
- Le seguenti variabili di ambiente relative alla creazione di client predefiniti sono diverse.

1.x	2.x
<code>AWS_CBOR_DISABLED</code>	<code>CBOR_ENABLED</code>

1.x	2.x
<code>AWS_ION_BINARY_DISABLE</code>	<code>BINARY_ION_ENABLED</code>

- Le seguenti proprietà di sistema relative alla creazione di client predefiniti sono diverse.

1.x	2.x
<code>com.amazonaws.sdk.disableEc2Metadata</code>	<code>aws.disableEc2Metadata</code>
<code>com.amazonaws.sdk.ec2MetadataServiceEndpointOverride</code>	<code>aws.ec2MetadataServiceEndpoint</code>
<code>com.amazonaws.sdk.disableCbor</code>	<code>aws.cborEnabled</code>
<code>com.amazonaws.sdk.disableIoBinary</code>	<code>aws.binaryIonEnabled</code>

- La versione 2.x non supporta le seguenti proprietà di sistema.

1.x
<code>com.amazonaws.sdk.disableCertChecking</code>
<code>com.amazonaws.sdk.enableDefaultMetrics</code>
<code>com.amazonaws.sdk.enableThrottledRetry</code>
<code>com.amazonaws.regions.RegionUtils.fileOverride</code>
<code>com.amazonaws.regions.RegionUtils.disableRemote</code>
<code>com.amazonaws.services.s3.disableImplicitGlobalClients</code>
<code>com.amazonaws.sdk.enableInRegionOptimizedMode</code>

- Il caricamento della configurazione della regione da un `endpoints.json` file personalizzato non è più supportato.

## Configurazione del client

Nella versione 1.x, la configurazione del client SDK è stata modificata impostando un'`ClientConfiguration`istanza sul client o sul client builder. Nella versione 2.x, la configurazione del client è suddivisa in classi di configurazione separate. Con le classi di configurazione separate, puoi configurare diversi client HTTP per client asincroni e sincroni, ma utilizzare comunque la stessa classe. `ClientOverrideConfiguration`

### Example della configurazione del client nella versione 1.x

```
AmazonDynamoDBClientBuilder.standard()  
    .withClientConfiguration(clientConfiguration)  
    .build()
```

### Example di configurazione del client sincrono nella versione 2.x

```
ProxyConfiguration.Builder proxyConfig = ProxyConfiguration.builder();  
  
ApacheHttpClient.Builder httpClientBuilder =  
    ApacheHttpClient.builder()  
        .proxyConfiguration(proxyConfig.build());  
  
ClientOverrideConfiguration.Builder overrideConfig =  
    ClientOverrideConfiguration.builder();  
  
DynamoDbClient client =  
    DynamoDbClient.builder()  
        .httpClientBuilder(httpClientBuilder)  
        .overrideConfiguration(overrideConfig.build())  
        .build();
```

### Example di configurazione del client asincrono nella versione 2.x

```
NettyNioAsyncHttpClient.Builder httpClientBuilder =  
    NettyNioAsyncHttpClient.builder();  
  
ClientOverrideConfiguration.Builder overrideConfig =  
    ClientOverrideConfiguration.builder();  
  
ClientAsyncConfiguration.Builder asyncConfig =  
    ClientAsyncConfiguration.builder();
```



```
DynamoDbAsyncClient client =
    DynamoDbAsyncClient.builder()
        .httpClientBuilder(httpClientBuilder)
        .overrideConfiguration(overrideConfig.build())
        .asyncConfiguration(asyncConfig.build())
        .build();
```

## Client HTTP

### Modifiche importanti

- Nella versione 2.x, è possibile modificare il client HTTP da utilizzare in fase di esecuzione specificando un'implementazione utilizzando `clientBuilder.httpClientBuilder`
- Quando si passa un client HTTP utilizzando `clientBuilder.httpClient` a un service client builder, il client HTTP non viene chiuso per impostazione predefinita se il client di servizio si chiude. Ciò consente di condividere client HTTP tra client di servizio.
- I client HTTP asincroni ora utilizzano un IO non bloccante.
- Alcune operazioni ora utilizzano HTTP/2 per migliorare le prestazioni.

### Modifiche alle impostazioni

Impostazione	1.x	Sincronizzazione 2.x, Apache	2.x Asincrono, Netty
	<pre>ClientCon figuration clientConfig =     new ClientCon figuration()</pre>	<pre>ApacheHtt pClient.B uilder httpClien tBuilder =     ApacheHtt pClient.b uilder()</pre>	<pre>NettyNioA syncHttpC lient.Builder httpClient tBuilder =     NettyNioA syncHttpC lient.builder()</pre>
Numero massimo di connessioni	<pre>clientCon fig.setMa xConnecti ons(...) clientCon fig.withM</pre>	<pre>httpClien tBuilder. maxConnec tions(...)</pre>	<pre>httpClien tBuilder. maxConcur rency(...)</pre>

Impostazione	1.x	Sincronizzazione 2.x, Apache	2.x Asincrono, Netty
	<code>axConnections(...)</code>		
Timeout di connessione	<code>clientConfig.setConnectionTimeout(...)</code> <code>clientConfig.withConnectionTimeout(...)</code>	<code>httpClientBuilder.connectionTimeout(...)</code> <code>httpClientBuilder.connectionAcquisitionTimeout(...)</code>	<code>httpClientBuilder.connectionTimeout(...)</code>
Timeout del socket	<code>clientConfig.setSocketTimeout(...)</code> <code>clientConfig.withSocketTimeout(...)</code>	<code>httpClientBuilder.socketTimeout(...)</code>	<code>httpClientBuilder.writeTimeout(...)</code> <code>httpClientBuilder.readTimeout(...)</code>
Connessione TTL	<code>clientConfig.setConnectionTTL(...)</code> <code>clientConfig.withConnectionTTL(...)</code>	<code>httpClientBuilder.connectionTimeToLive(...)</code>	<code>httpClientBuilder.connectionTimeToLive(...)</code>

Impostazione	1.x	Sincronizzazione 2.x, Apache	2.x Asincrono, Netty
Connessione massima inattiva	<pre>clientConfig.setConnectionMaxIdleMillis(...) clientConfig.withConnectionMaxIdleMillis(...)</pre>	<pre>httpClientBuilder.connectionMaxIdleTime(...)</pre>	<pre>httpClientBuilder.connectionMaxIdleTime(...)</pre>
Convalida dopo l'inattività	<pre>clientConfig.setValidateAfterInactivityMillis(...) clientConfig.withValidateAfterInactivityMillis(...)</pre>	Non supportato (funzionalità di <a href="#">richiesta</a> )	Non supportato ( <a href="#">funzionalità di richiesta</a> )
Indirizzo locale	<pre>clientConfig.setLocalAddress(...) clientConfig.withLocalAddress(...)</pre>	<pre>httpClientBuilder.localAddress(...)</pre>	<a href="#">Non supportato</a>
Expect-continue abilitato	<pre>clientConfig.setUseExpectContinue(...) clientConfig.withUseExpectContinue(...)</pre>	<pre>httpClientBuilder.expectContinueEnabled(...)</pre>	Non supportato (funzionalità di <a href="#">richiesta</a> )

Impostazione	1.x	Sincronizzazione 2.x, Apache	2.x Asincrono, Netty
Connection Reaper	<pre>clientConfig.setUseReaper(...) clientConfig.withReaper(...)</pre>	<pre>httpClientBuilder.useIdleConnectionReaper(...)</pre>	<pre>httpClientBuilder.useIdleConnectionReaper(...)</pre>
	<pre>AmazonDynamoDBClientBuilder .standard() .withClientConfiguration(clientConfiguration) .build()</pre>	<pre>DynamoDBClient.builder() .httpClientBuilder(httpClientBuilder) .build()</pre>	<pre>DynamoDBAsyncClient.builder() .httpClientBuilder(httpClientBuilder) .build()</pre>

## Proxy client HTTP

Impostazioni	1.x	Sincronizzazione 2.x, Apache	2.x Asincrono, Netty
	<pre>ClientConfiguration clientConfig = new ClientConfiguration()</pre>	<pre>ProxyConfiguration .Builder proxyConfig = ProxyConfiguration .builder()</pre>	<pre>ProxyConfiguration .Builder proxyConfig = ProxyConfiguration .builder()</pre>
Host proxy	<pre>clientConfig.setProxyHost(...)</pre>	<pre>proxyConfig.endpoint(...)</pre>	<pre>proxyConfig.host(...)</pre>

Impostazioni	1.x	Sincronizzazione 2.x, Apache	2.x Asincrono, Netty
	<code>clientConfig.withProxyHost(...)</code>		
Porta proxy	<code>clientConfig.setProxyPort(...)</code> <code>clientConfig.withProxyPort(...)</code>	<code>proxyConfig.endpoint(...)</code>  La <a href="#">porta proxy</a> è incorporata in endpoint	<code>proxyConfig.port(...)</code>
Nome utente proxy	<code>clientConfig.setProxyUsername(...)</code> <code>clientConfig.withProxyUsername(...)</code>	<code>proxyConfig.username(...)</code>	<code>proxyConfig.username(...)</code>
Password proxy	<code>clientConfig.setProxyPassword(...)</code> <code>clientConfig.withProxyPassword(...)</code>	<code>proxyConfig.password(...)</code>	<code>proxyConfig.password(...)</code>
Dominio proxy	<code>clientConfig.setProxyDomain(...)</code> <code>clientConfig.withProxyDomain(...)</code>	<code>proxyConfig.ntlmDomain(...)</code>	Non supportato ( <a href="#">funzionalità di richiesta</a> )

Impostazioni	1.x	Sincronizzazione 2.x, Apache	2.x Asincrono, Netty
Workstation proxy	<pre>clientConfig.setProxyWorkspace(...) clientConfig.withProxyWorkstation(...)</pre>	<pre>proxyConfig.ntlmWorkstation(...)</pre>	Non supportata ( <a href="#">funzionalità di richiesta</a> )
Metodi di autenticazione proxy	<pre>clientConfig.setProxyAuthenticationMethods(...) clientConfig.withProxyAuthenticationMethods(...)</pre>	<a href="#">Non supportato</a>	Non supportato ( <a href="#">funzionalità di richiesta</a> )
Autenticazione proxy di base preventiva	<pre>clientConfig.setPreemptiveBasicProxyAuth(...) clientConfig.withPreemptiveBasicProxyAuth(...)</pre>	<pre>proxyConfig.preemptiveBasicAuthenticationEnabled(...)</pre>	Non supportata ( <a href="#">funzionalità di richiesta</a> )

Impostazioni	1.x	Sincronizzazione 2.x, Apache	2.x Asincrono, Netty
Host non proxy	<pre>clientConfig.setNonProxyHosts(...) clientConfig.withNonProxyHosts(...)</pre>	<pre>proxyConfiguration.nonProxyHosts(...)</pre>	<pre>proxyConfiguration.nonProxyHosts(...)</pre>
Disabilita il socket proxy	<pre>clientConfig.setDisableSocketProxy(...) clientConfig.withDisableSocketProxy(...)</pre>	Non supportato ( <a href="#">funzionalità di richiesta</a> )	Non supportato ( <a href="#">funzionalità di richiesta</a> )
	<pre>AmazonDynamoDBClientBuilder.standard()     .withClientConfiguration(clientConfiguration)     .build()</pre>	<pre>httpClientBuilder.proxyConfiguration(proxyConfiguration).build()</pre>	<pre>httpClientBuilder.proxyConfiguration(proxyConfiguration).build()</pre>

## Sostituzioni del client

Impostazione	1.x	2.x
	<pre>ClientConfiguration clientConfig =     new ClientCon figuration()</pre>	<pre>ClientOverrideConf figuration.Builder overrideConfig =     ClientOverrideConf figuration.builder()</pre>
Prefisso dell'agente utente	<pre>clientConfig.setUs erAgentPrefix(...) clientConfig.with UserAgentPrefix(...)</pre>	<pre>overrideConfig.adv ancedOption(     SdkAdvancedClientO ption.USER_AGENT_P REFIX, ...)</pre>
Suffisso dell'agente utente	<pre>clientConfig.setUs erAgentSuffix(...) clientConfig.with UserAgentSuffix(...)</pre>	<pre>overrideConfig.adv ancedOption(     SdkAdvancedClientO ption.USER_AGENT_S UFFIX, ...)</pre>
Signer	<pre>clientConfig.setSi gnerOverride(...) clientConfig.withS ignerOverride(...)</pre>	<pre>overrideConfig.adv ancedOption(     SdkAdvancedClientO ption.SIGNER, ...)</pre>
Intestazioni aggiuntive	<pre>clientConfig.addHe ader(...) clientConfig.with Header(...)</pre>	<pre>overrideConfig.put Header(...)</pre>
Timeout richiesta	<pre>clientConfig.setRe questTimeout(...) clientConfig.withR equestTimeout(...)</pre>	<pre>overrideConfig.api CallAttemptTimeout (...)</pre>



Impostazione	1.x	2.x
Timeout di esecuzione del client	<pre>clientConfig.setClientExecutionTimeout(...) clientConfig.withClientExecutionTimeout(...)</pre>	<pre>overrideConfig.apiCallTimeout(...)</pre>
Usa Gzip	<pre>clientConfig.setUseGzip(...) clientConfig.withGzip(...)</pre>	Non supportato ( <a href="#">funzionalità di richiesta</a> )
Suggerimento sulla dimensione e del buffer del socket	<pre>clientConfig.setSocketBufferSizeHints(...) clientConfig.withSocketBufferSizeHints(...)</pre>	Non supportato (funzionalità di <a href="#">richiesta</a> )
Metadati di risposta alla cache	<pre>clientConfig.setCacheResponseMetadata(...) clientConfig.withCacheResponseMetadata(...)</pre>	Non supportato ( <a href="#">funzionalità di richiesta</a> )
Dimensione della cache dei metadati di risposta	<pre>clientConfig.setResponseMetadataCacheSize(...) clientConfig.withResponseMetadataCacheSize(...)</pre>	Non supportata ( <a href="#">funzionalità di richiesta</a> )

Impostazione	1.x	2.x
resolver DNS	<pre>clientConfig.setDnsResolver(...) clientConfig.withDnsResolver(...)</pre>	Non supportato ( <a href="#">funzionalità di richiesta</a> )
TCP keepalive	<pre>clientConfig.setUseTcpKeepAlive(...) clientConfig.withTcpKeepAlive(...)</pre>	<p>Questa opzione è ora nella configurazione del client HTTP</p> <ul style="list-style-type: none"> <li>- <code>ApacheHttpClient.builder().tcpKeepAlive(true)</code></li> <li>- <code>NettyNioAsyncHttpClient.builder().tcpKeepAlive(true)</code></li> </ul>
Secure Random	<pre>clientConfig.setSecureRandom(...) clientConfig.withSecureRandom(...)</pre>	Non supportato ( <a href="#">funzionalità di richiesta</a> )
	<pre>AmazonDynamoDBClientBuilder.standard()     .withClientConfiguration(clientConfiguration)     .build()</pre>	<pre>DynamoDbClient.builder()     .httpClientBuilder(httpClientBuilder)     .build()</pre>

## Il client annulla i tentativi

Impostazione	1.x	2.x
	<pre>ClientConfiguration clientConfig =</pre>	<pre>RetryPolicy.Builder retryPolicy =</pre>

Impostazione	1.x	2.x
	<pre>new ClientConfiguration()</pre>	<pre>RetryPolicy.builder()</pre>
Numero massimo di tentativi di errore	<pre>clientConfig.setMaxErrorRetry(...) clientConfig.withMaxErrorRetry(...)</pre>	<pre>retryPolicy.numRetries(...)</pre>
Usa tentativi limitati	<pre>clientConfig.setUseThrottleRetries(...) clientConfig.withUseThrottleRetries(...)</pre>	<a href="#">Non supportato</a>
Numero massimo di tentativi consecutivi prima della limitazione	<pre>clientConfig.setMaxConsecutiveRetriesBeforeThrottling(...) clientConfig.withMaxConsecutiveRetriesBeforeThrottling(...)</pre>	<a href="#">Non supportato</a>
	<pre>AmazonDynamoDBClientBuilder.standard()     .withClientConfiguration(clientConfiguration)     .build()</pre>	<pre>DynamoDbClient.builder()     .httpClientBuilder(httpClientBuilder)     .build()</pre>

## Client asincroni

Impostazione	1.x	2.x
		<pre>ClientAsyncConfiguration.Builder asyncConfig =     ClientAsyncConfiguration.builder()</pre>
Esecutore	<pre>AmazonDynamoDBAsyncClientBuilder.standard()     .withExecutorFactory(...)     .build()</pre>	<pre>asyncConfig.advancedOption(     SdkAdvancedAsyncClientOption.FUTURE_COMPLETION_EXECUTOR, ...)</pre>
		<pre>DynamoDbAsyncClient.builder()     .asyncConfiguration(asyncConfig)     .build()</pre>

## Altre modifiche al cliente

La seguente `ClientConfiguration` opzione di 1.x è stata modificata nella versione 2.x dell'SDK e non ha un equivalente diretto.

Impostazione	1.x	Equivalente a 2.x
Protocollo	<pre>clientConfig.setProtocol(Protocol.HTTP) clientConfig.withProtocol(Protocol.HTTP)</pre>	<p>L'impostazione predefinita del protocollo è HTTPS. Per modificare l'impostazione, specifica il protocollo che imposta un endpoint HTTP nel client builder:</p>

Impostazione	1.x	Equivalente a 2.x
		<pre>clientBuilder.endpointOverride(     URI.create("http://..."))</pre>

## Modifiche al provider delle credenziali

Questa sezione fornisce una mappatura delle modifiche dei nomi delle classi e dei metodi del provider di credenziali tra le versioni 1.x e 2.x di AWS SDK for Java

### Differenze notevoli

- In 2.x, il provider di credenziali predefinito carica le proprietà del sistema prima delle variabili d'ambiente. Per ulteriori informazioni, vedere [Utilizzo delle credenziali](#).
- Il metodo del costruttore è sostituito dai metodi `create` o `builder`.

#### Example

```
DefaultCredentialsProvider.create();
```

- L'aggiornamento asincrono non è più configurato per impostazione predefinita e deve essere specificato con il `builder` del provider di credenziali.

#### Example

```
ContainerCredentialsProvider provider = ContainerCredentialsProvider.builder()
    .asyncCredentialUpdateEnabled(true)
    .build();
```

- È possibile indicare il percorso di un file di profilo personalizzato utilizzando `ProfileCredentialsProvider.builder()`.

#### Example

```
ProfileCredentialsProvider profile = ProfileCredentialsProvider.builder()
    .profileFile(ProfileFile.builder().content(Paths.get("myProfileFile.file")).build())
```

```
.build();
```

- Il formato del file di profilo è stato modificato per migliorare la corrispondenza con la AWS CLI. Per i dettagli, vedere [Configurazione di AWS CLI nella Guida](#) per l'AWS Command Line Interface.

## Modifiche al provider di credenziali mappate tra le versioni 1.x e 2.x

### AWSCredentialsProvider

Cambia categoria	1.x	2.x
Nome del pacchetto/classe	<code>com.amazonaws.auth.AWSCredentialsProvider</code>	<code>software.amazon.awssdk.auth.credentials.AwsCredentialsProvider</code>
Nome del metodo	<code>getCredentials</code>	<code>resolveCredentials</code>
Metodo non supportato	<code>refresh</code>	Non supportato

### DefaultAWSCredentialsProviderChain

Cambia categoria	1.x	2.x
Nome del pacchetto/classe	<code>com.amazonaws.auth.DefaultAWSCredentialsProviderChain</code>	<code>software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider</code>
Creazione	<code>new DefaultAWSCredentialsProviderChain</code>	<code>DefaultCredentialsProvider.create</code>
Metodo non supportato	<code>getInstance</code>	Non supportato
Ordine di priorità delle impostazioni esterne	Le variabili di ambiente prima delle proprietà del sistema	Le proprietà del sistema prima delle variabili di ambiente

**AWSStaticCredentialsProvider**

Cambia categoria	1.x	2.x
Nome del pacchetto/classe	<code>com.amazonaws.auth.AWSStaticCredentialsProvider</code>	<code>software.amazon.awssdk.auth.credentials.StaticCredentialsProvider</code>
Creazione	<code>new AWSStaticCredentialsProvider</code>	<code>StaticCredentialsProvider.create</code>

**EnvironmentVariableCredentialsProvider**

Cambia categoria	1.x	2.x
Nome del pacchetto/classe	<code>com.amazonaws.auth.EnvironmentVariableCredentialsProvider</code>	<code>software.amazon.awssdk.auth.credentials.EnvironmentVariableCredentialsProvider</code>
Creazione	<code>new EnvironmentVariableCredentialsProvider</code>	<code>EnvironmentVariableCredentialsProvider.create</code>
Nome della variabile di ambiente	<code>AWS_ACCESS_KEY</code>	<code>AWS_ACCESS_KEY_ID</code>
	<code>AWS_SECRET_KEY</code>	<code>AWS_SECRET_ACCESS_KEY</code>

## SystemPropertiesCredentialsProvider

Cambia categoria	1.x	2.x
Nome del pacchetto/classe	<code>com.amazonaws.auth. SystemPropertiesC redentialsProvider</code>	<code>software.amazon.aw ssdk.auth.credenti als.SystemProperty CredentialsProvider</code>
Creazione	<code>new SystemPro pertiesCredentials Provider</code>	<code>SystemPropertiesCr edentialsProvider. create</code>
Nome della proprietà del sistema	<code>aws.secretKey</code>	<code>aws.secretAccessKey</code>

## ProfileCredentialsProvider

Cambia categoria	1.x	2.x
Nome del pacchetto/classe	<code>com.amazonaws.auth .profile.ProfileCr edentialsProvider</code>	<code>software.amazon.aw ssdk.auth.credenti als.ProfileCredent ialsProvider</code>
Creazione	<code>new ProfileCr edentialsProvider</code>	<code>ProfileCredentials Provider.create</code>
Ubicazione del profilo personalizzato	<ul style="list-style-type: none"> <li>• Variabile di ambiente <code>AWS_CREDENTIAL_PROFILES_FILE</code></li> <li>• <code>new ProfileCr edentialsProvider</code></li> </ul>	<ul style="list-style-type: none"> <li>• Variabile di ambiente <code>AWS_SHARED_CREDENTIALS_FILE</code></li> <li>• <code>ProfileCredentials Provider.builder</code></li> </ul>



**ContainerCredentialsProvider**

Cambia categoria	1.x	2.x
Nome del pacchetto/classe	<code>com.amazonaws.auth.ContainerCredentialsProvider</code>	<code>software.amazon.awssdk.auth.credentials.ContainerCredentialsProvider</code>
Creazione	<code>new ContainerCredentialsProvider</code>	<code>ContainerCredentialsProvider.create</code>
Specificare l'aggiornamento asincrono	Non supportato	Comportamento predefinito

**InstanceProfileCredentialsProvider**

Cambia categoria	1.x	2.x
Nome del pacchetto/classe	<code>com.amazonaws.auth.InstanceProfileCredentialsProvider</code>	<code>software.amazon.awssdk.auth.credentials.InstanceProfileCredentialsProvider</code>
Creazione	<code>new InstanceProfileCredentialsProvider</code>	<code>InstanceProfileCredentialsProvider.create</code>
Specificare l'aggiornamento asincrono	<code>new InstanceProfileCredentialsProvider(true)</code>	<code>InstanceProfileCredentialProvider.builder().asyncCredentialUpdateEnabled(true).build()</code>
Nome della proprietà di sistema	<code>com.amazonaws.sdk.disableEc2Metadata</code>	<code>aws.disableEc2Metadata</code>

Cambia categoria	1.x	2.x
	<code>com.amazonaws.sdk.ec2MetadataServiceEndpointOverride</code>	<code>aws.ec2MetadataServiceEndpoint</code>

## STSAssumeRoleSessionCredentialsProvider

Cambia categoria	1.x	2.x
Nome del pacchetto/classe	<code>com.amazonaws.auth.STSAssumeRoleSessionCredentialsProvider</code>	<code>software.amazon.awssdk.services.sts.auth.StsAssumeRoleCredentialsProvider</code>
Creazione	<ul style="list-style-type: none"> <li><code>new STSAssumeRoleSessionCredentialsProvider</code></li> <li><code>new STSAssumeRoleSessionCredentialsProvider.Builder</code></li> </ul>	<code>StsAssumeRoleCredentialsProvider.builder</code>
Aggiornamento asincrono	Comportamento predefinito	Comportamento predefinito
Configurazione	<code>new STSAssumeRoleSessionCredentialsProvider.Builder</code>	Configura una richiesta e <code>StsClient AssumeRoleRequest</code>

**STSSessionCredentialsProvider**

Cambia categoria	1.x	2.x
Nome del pacchetto/classe	<code>com.amazonaws.auth.STSSessionCredentialsProvider</code>	<code>software.amazon.awssdk.services.sts.auth.StsGetSessionTokenCredentialsProvider</code>
Creazione	<code>new STSAssumeRoleSessionCredentialsProvider</code>	<code>StsGetSessionTokenCredentialsProvider.builder</code>
Aggiornamento asincrono	Comportamento predefinito	<code>StsGetSessionTokenCredentialsProvider.builder</code>
Configurazione	Parametri del costruttore	Configura una <code>GetSessionTokenRequest</code> richiesta <code>StsClient</code> and in un builder

**WebIdentityFederationSessionCredentialsProvider**

Cambia categoria	1.x	2.x
Nome del pacchetto/classe	<code>com.amazonaws.auth.WebIdentityFederationSessionCredentialsProvider</code>	<code>software.amazon.awssdk.services.sts.auth.StsAssumeRoleWithWebIdentityCredentialsProvider</code>
Creazione	<code>new WebIdentityFederationSessionCredentialsProvider</code>	<code>StsAssumeRoleWithWebIdentityCredentialsProvider.builder</code>

Cambia categoria	1.x	2.x
Aggiornamento asincrono	Comportamento predefinito	<code>StsAssumeRoleWithWebIdentityCredentialsProvider.builder</code>
Configurazione	Parametri del costruttore	Configura una <code>AssumeRoleWithWebIdentityRequest</code> richiesta <code>StsClient</code> and in un builder

### Classi sostituite

classe 1.x	classi sostitutive 2.x
<code>com.amazonaws.auth.EC2ContainerCredentialsProviderWrapper</code>	<code>software.amazon.awssdk.auth.credentials.ContainerCredentialsProvider</code> e <code>software.amazon.awssdk.auth.credentials.InstanceProfileCredentialsProvider</code>
<code>com.amazonaws.services.s3.S3CredentialsProviderChain</code>	<code>software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider</code> e <code>software.amazon.awssdk.auth.credentials.AnonymousCredentialsProvider</code>

### Classi rimosse

classe 1.x
<code>com.amazonaws.auth.ClasspathPropertiesFileCredentialsProvider</code>
<code>com.amazonaws.auth.PropertiesFileCredentialsProvider</code>

## Modifiche alla regione

Questa sezione descrive le modifiche implementate nella versione AWS SDK for Java 2.x per l'utilizzo delle classi `and. Region` `Regions`

### Configurazione della regione

- Alcuni AWS servizi non dispongono di endpoint specifici per regione. Quando si utilizzano questi servizi, è necessario impostare la regione come `Region.AWS_GLOBAL` o `Region.AWS_CN_GLOBAL`.

#### Example

```
Region region = Region.AWS_GLOBAL;
```

- Le classi `com.amazonaws.regions.Regions` e `com.amazonaws.regions.Region` sono ora combinate in un'unica classe, `software.amazon.awssdk.regions.Region`.

### Mappature dei nomi di metodi e classi

Le tabelle seguenti mappano le classi relative alla regione tra le versioni 1.x e 2.x di AWS SDK for Java. È possibile creare un'istanza di tali classi utilizzando il metodo `of()`.

#### Example

```
RegionMetadata regionMetadata = RegionMetadata.of(Region.US_EAST_1);
```

#### 1.x Modifiche al metodo della classe `Regions`

1.x	2.x
<code>Regions.fromName</code>	<code>Region.of</code>
<code>Regions.getName</code>	<code>Region.id</code>
<code>Regions.getDescription</code>	<code>Region.metadata().description()</code>
<code>Regions.getCurrentRegion</code>	Non supportato
<code>Regions.DEFAULT_REGION</code>	Non supportato

1.x	2.x
<code>Regions.name</code>	<code>Region.id</code>

### 1.x Modifiche al metodo della classe Region

1.x	2.x
<code>Region.getName</code>	<code>Region.id</code>
<code>Region.hasHttpsEndpoint</code>	Non supportato
<code>Region.hasHttpEndpoint</code>	Non supportato
<code>Region.getAvailableEndpoints</code>	Non supportato
<code>Region.createClient</code>	Non supportato

### RegionMetadata modifiche al metodo di classe

1.x	2.x
<code>RegionMetadata.getName</code>	<code>RegionMetadata.name</code>
<code>RegionMetadata.getDomain</code>	<code>RegionMetadata.domain</code>
<code>RegionMetadata.getPartition</code>	<code>RegionMetadata.partition</code>

### ServiceMetadata modifiche al metodo di classe

1.x	2.x
<code>Region.getServiceEndpoint</code>	<code>ServiceMetadata.endpointFor(Region)</code>
<code>Region.isServiceSupported</code>	<code>ServiceMetadata.regions().contains(Region)</code>

## Modifiche alle operazioni, alle richieste e alle risposte

Nella versione 2.x dell'SDK for Java, le richieste vengono passate a un'operazione client. Ad esempio `DynamoDbClient`'s `PutItemRequest` viene passato all'operazione. `DynamoDbClient.putItem`. Queste operazioni restituiscono una risposta da Servizio AWS, ad esempio un `PutItemResponse`.

La versione 2.x dell'SDK for Java presenta le seguenti modifiche rispetto alla 1.x.

- Le operazioni con più pagine di risposta ora dispongono di un `Paginator` metodo per iterare automaticamente su tutti gli elementi della risposta.
- Non è possibile modificare le richieste e le risposte.
- È necessario creare richieste e risposte con un metodo di creazione statico anziché con un costruttore. Ad esempio, 1.x's è ora. `new PutItemRequest().withTableName(...)`  
`PutItemRequest.builder().tableName(...).build()`
- Le operazioni supportano un modo breve per creare richieste:  
`dynamoDbClient.putItem(request -> request.tableName(...))`

## Operazioni di streaming

Le operazioni di streaming come Amazon S3 `getObject` e `putObject` i metodi ora supportano I/O non bloccanti. Di conseguenza, i POJO di richiesta e risposta non accettano più un come parametro. `InputStream` Per le richieste sincrone, invece, l'oggetto della richiesta accetta `RequestBody`, che è un flusso di byte. L'equivalente asincrono accetta un `AsyncRequestBody`

Example del **putObject** funzionamento di Amazon S3 nella versione 1.x

```
s3client.putObject(BUCKET, KEY, new File(file_path));
```

Example del **putObject** funzionamento di Amazon S3 nella versione 2.x

```
s3client.putObject(PutObjectRequest.builder()  
    .bucket(BUCKET)  
    .key(KEY)  
    .build(),  
    RequestBody.of(Paths.get("myfile.in")));
```

In parallelo, un oggetto di risposta in streaming accetta un `ResponseTransformer` per i client sincroni e un `AsyncResponseTransformer` per i client asincroni.

## Example del **getObject** funzionamento di Amazon S3 nella versione 1.x

```
S3Object o = s3.getObject(bucket, key);
S3ObjectInputStream s3is = o.getObjectContent();
FileOutputStream fos = new FileOutputStream(new File(key));
```

## Example del **getObject** funzionamento di Amazon S3 nella versione 2.x

```
s3client.getObject(GetObjectRequest.builder().bucket(bucket).key(key).build(),
    ResponseTransformer.toFile(Paths.get("key")));
```

Nell'SDK for Java 2.x, le operazioni di risposta in streaming dispongono di `AsBytes` un metodo per caricare la risposta in memoria e semplificare le comuni conversioni di tipo in memoria.

## Modifiche alle eccezioni

I nomi delle classi di eccezione, le loro strutture e le loro relazioni sono cambiati.

`software.amazon.awssdk.core.exception.SdkException` è la nuova `Exception` classe base estesa da tutte le altre eccezioni.

La tabella fornisce una mappatura delle modifiche ai nomi delle classi di eccezioni.

1.x	2.x
<code>com.amazonaws.SdkBaseException</code> <code>com.amazonaws.AmazonClientException</code>	<code>software.amazon.awssdk.core.exception.SdkException</code>
<code>com.amazonaws.SdkClientException</code>	<code>software.amazon.awssdk.core.exception.SdkClientException</code>
<code>com.amazonaws.AmazonServiceException</code>	<code>software.amazon.awssdk.awscore.exception.AwsServiceException</code>

La tabella seguente mappa i metodi sulle classi di eccezioni tra le versioni 1.x e 2.x.



1.x	2.x
<code>AmazonServiceException.getRequestId</code>	<code>SdkServiceException.requestId</code>
<code>AmazonServiceException.getServiceName</code>	<code>AwsServiceException.awsErrorDetails().serviceName</code>
<code>AmazonServiceException.getErrorCode</code>	<code>AwsServiceException.awsErrorDetails().errorCode</code>
<code>AmazonServiceException.getErrorMessage</code>	<code>AwsServiceException.awsErrorDetails().errorMessage</code>
<code>AmazonServiceException.getStatusCode</code>	<code>AwsServiceException.awsErrorDetails().sdkHttpResponse().statusCode</code>
<code>AmazonServiceException.getHttpHeaders</code>	<code>AwsServiceException.awsErrorDetails().sdkHttpResponse().headers</code>
<code>AmazonServiceException.getRawResponse</code>	<code>AwsServiceException.awsErrorDetails().rawResponse</code>

## Modifiche alla serializzazione

Gli SDK for Java v1.x e v2.x differiscono nel modo in cui serializzano gli oggetti List per richiedere i parametri.

L'SDK for Java 1.x non serializza un elenco vuoto, mentre l'SDK for Java 2.x serializza un elenco vuoto come parametro vuoto.

Ad esempio, considera un servizio con `a` che richiede un `SampleOperation` `SampleRequest` `SampleRequest` accetta due parametri, un tipo `String str1` e un tipo `ListListParam`, come mostrato negli esempi seguenti.

## Example **SampleOperation**o in 1.x

```
SampleRequest v1Request = new SampleRequest()
    .withStr1("TestName");

sampleServiceV1Client.sampleOperation(v1Request);
```

La registrazione a livello di cavo mostra che il `listParam` parametro non è serializzato.

```
Action=SampleOperation&Version=2011-01-01&str1=TestName
```

## Example **SampleOperation**di in 2.x

```
sampleServiceV2Client.sampleOperation(b -> b
    .str1("TestName"));
```

La registrazione a livello di cavo mostra che il `listParam` parametro è serializzato senza alcun valore.

```
Action=SampleOperation&Version=2011-01-01&str1=TestName&listParam=
```

## Modifiche specifiche del servizio

### Modifiche ad Amazon S3

SDK for Java 2.x disabilita l'accesso anonimo per impostazione predefinita. Di conseguenza, è necessario abilitare l'accesso anonimo utilizzando `AnonymousCredentialsProvider`

### Modifiche al nome dell'operazione

Molti dei nomi delle operazioni per il Amazon S3 client sono cambiati nella versione AWS SDK for Java 2.x. Nella versione 1.x, il client Amazon S3 non viene generato direttamente dall'API di servizio. Questo provocava incoerenze tra l'API di servizio e le operazioni SDK. Nella versione 2.x, il client Amazon S3 viene generato in modo da garantire una maggiore coerenza con l'API di servizio.

La tabella seguente mostra i nomi delle operazioni nelle due versioni.

## Nomi delle operazioni di Amazon S3

1.x	2.x
<code>abortMultipartUpload</code>	<code>abortMultipartUpload</code>
<code>changeObjectStorageClass</code>	<code>copyObject</code>
<code>completeMultipartUpload</code>	<code>completeMultipartUpload</code>
<code>copyObject</code>	<code>copyObject</code>
<code>copyPart</code>	<code>uploadPartCopy</code>
<code>createBucket</code>	<code>createBucket</code>
<code>deleteBucket</code>	<code>deleteBucket</code>
<code>deleteBucketAnalyticsConfiguration</code>	<code>deleteBucketAnalyticsConfiguration</code>
<code>deleteBucketCrossOriginConfiguration</code>	<code>deleteBucketCors</code>
<code>deleteBucketEncryption</code>	<code>deleteBucketEncryption</code>
<code>deleteBucketInventoryConfiguration</code>	<code>deleteBucketInventoryConfiguration</code>
<code>deleteBucketLifecycleConfiguration</code>	<code>deleteBucketLifecycle</code>
<code>deleteBucketMetricsConfiguration</code>	<code>deleteBucketMetricsConfiguration</code>
<code>deleteBucketPolicy</code>	<code>deleteBucketPolicy</code>
<code>deleteBucketReplicationConfiguration</code>	<code>deleteBucketReplication</code>
<code>deleteBucketTaggingConfiguration</code>	<code>deleteBucketTagging</code>
<code>deleteBucketWebsiteConfiguration</code>	<code>deleteBucketWebsite</code>

1.x	2.x
<code>deleteObject</code>	<code>deleteObject</code>
<code>deleteObjectTagging</code>	<code>deleteObjectTagging</code>
<code>deleteObjects</code>	<code>deleteObjects</code>
<code>deleteVersion</code>	<code>deleteObject</code>
<code>disableRequesterPays</code>	<code>putBucketRequestPayment</code>
<code>doesBucketExist</code>	<code>headBucket</code>
<code>doesBucketExistV2</code>	<code>headBucket</code>
<code>doesObjectExist</code>	<code>headObject</code>
<code>enableRequesterPays</code>	<code>putBucketRequestPayment</code>
<code>generatePresignedUrl</code>	<a href="#">S3Presigner</a>
<code>getBucketAccelerateConfiguration</code>	<code>getBucketAccelerateConfiguration</code>
<code>getBucketAcl</code>	<code>getBucketAcl</code>
<code>getBucketAnalyticsConfiguration</code>	<code>getBucketAnalyticsConfiguration</code>
<code>getBucketCrossOriginConfiguration</code>	<code>getBucketCors</code>
<code>getBucketEncryption</code>	<code>getBucketEncryption</code>
<code>getBucketInventoryConfiguration</code>	<code>getBucketInventoryConfiguration</code>
<code>getBucketLifecycleConfiguration</code>	<code>getBucketLifecycle</code> o <code>getBucketLifecycleConfiguration</code>
<code>getBucketLocation</code>	<code>getBucketLocation</code>
<code>getBucketLoggingConfiguration</code>	<code>getBucketLogging</code>

1.x	2.x
<code>getBucketMetricsConfiguration</code>	<code>getBucketMetricsConfiguration</code>
<code>getBucketNotificationConfiguration</code>	<code>getBucketNotification</code> o <code>getBucketNotificationConfiguration</code>
<code>getBucketPolicy</code>	<code>getBucketPolicy</code>
<code>getBucketReplicationConfiguration</code>	<code>getBucketReplication</code>
<code>getBucketTaggingConfiguration</code>	<code>getBucketTagging</code>
<code>getBucketVersioningConfiguration</code>	<code>getBucketVersioning</code>
<code>getBucketWebsiteConfiguration</code>	<code>getBucketWebsite</code>
<code>getObject</code>	<code>getObject</code>
<code>getObjectAcl</code>	<code>getObjectAcl</code>
<code>getObjectAsString</code>	<code>getObjectAsBytes().asUtf8String</code>
<code>getObjectMetadata</code>	<code>headObject</code>
<code>getObjectTagging</code>	<code>getObjectTagging</code>
<code>getResourceUrl</code>	<a href="#">S3Utilities#getUrl</a>
<code>getS3AccountOwner</code>	<code>listBuckets</code>
<code>getUrl</code>	<a href="#">S3Utilities#getUrl</a>
<code>headBucket</code>	<code>headBucket</code>
<code>initiateMultipartUpload</code>	<code>createMultipartUpload</code>
<code>isRequesterPaysEnabled</code>	<code>getBucketRequestPayment</code>

1.x	2.x
<code>listBucketAnalyticsConfigurations</code>	<code>listBucketAnalyticsConfigurations</code>
<code>listBucketInventoryConfigurations</code>	<code>listBucketInventoryConfigurations</code>
<code>listBucketMetricsConfigurations</code>	<code>listBucketMetricsConfigurations</code>
<code>listBuckets</code>	<code>listBuckets</code>
<code>listMultipartUploads</code>	<code>listMultipartUploads</code>
<code>listNextBatchOfObjects</code>	<code>listObjectsV2Paginator</code>
<code>listNextBatchOfVersions</code>	<code>listObjectVersionsPaginator</code>
<code>listObjects</code>	<code>listObjects</code>
<code>listObjectsV2</code>	<code>listObjectsV2</code>
<code>listParts</code>	<code>listParts</code>
<code>listVersions</code>	<code>listObjectVersions</code>
<code>putObject</code>	<code>putObject</code>
<code>restoreObject</code>	<code>restoreObject</code>
<code>restoreObjectV2</code>	<code>restoreObject</code>
<code>selectObjectContent</code>	<code>selectObjectContent</code>
<code>setBucketAccelerateConfiguration</code>	<code>putBucketAccelerateConfiguration</code>
<code>setBucketAcl</code>	<code>putBucketAcl</code>
<code>setBucketAnalyticsConfiguration</code>	<code>putBucketAnalyticsConfiguration</code>

1.x	2.x
setBucketCrossOriginConfiguration	putBucketCors
setBucketEncryption	putBucketEncryption
setBucketInventoryConfiguration	putBucketInventoryConfiguration
setBucketLifecycleConfiguration	putBucketLifecycle o putBucketLifecycleConfiguration
setBucketLoggingConfiguration	putBucketLogging
setBucketMetricsConfiguration	putBucketMetricsConfiguration
setBucketNotificationConfiguration	putBucketNotification o putBucketNotificationConfiguration
setBucketPolicy	putBucketPolicy
setBucketReplicationConfiguration	putBucketReplication
setBucketTaggingConfiguration	putBucketTagging
setBucketVersioningConfiguration	putBucketVersioning
setBucketWebsiteConfiguration	putBucketWebsite
setObjectAcl	putObjectAcl
setObjectRedirectLocation	copyObject
setObjectTagging	putObjectTagging
uploadPart	uploadPart

## Modifiche ad Amazon SNS

Un client SNS non può più accedere agli argomenti SNS in regioni diverse dalla regione per cui è configurato.

## Modifiche ad Amazon SQS

Un client SQS non può più accedere alle code SQS in regioni diverse dalla regione a cui è configurato per accedere.

## Modifiche ad Amazon RDS

L'SDK for Java 2.x `RdsUtilities#generateAuthenticationToken` utilizza al posto della `RdsIamAuthTokenGenerator` classe in 1.x.

## Modifiche al file di profilo

AWS SDK for Java 2.x Analizza le definizioni dei profili `~/.aws/config` e emula più `~/.aws/credentials` da vicino il modo in cui la AWS CLI analizza i file.

L'SDK per Java 2.x:

- Risolve un `~/` OR `~` seguito dal separatore di percorso predefinito del file system all'inizio del percorso controllando, nell'ordine,, (solo Windows)`$HOME`, `$USERPROFILE` (solo Windows) e quindi la proprietà del sistema. `$HOMEDRIVE` `$HOMEPATH` `user.home`
- Cerca la variabile di `AWS_SHARED_CREDENTIALS_FILE` ambiente anziché `AWS_CREDENTIAL_PROFILES_FILE`
- Elimina silenziosamente le definizioni dei profili nei file di configurazione senza la parola `profile` all'inizio del nome del profilo.
- Elimina silenziosamente le definizioni di profilo che non sono costituite da caratteri alfanumerici, trattini bassi o trattini (dopo che la `profile` parola principale è stata rimossa dai file di configurazione).
- Unisce le impostazioni delle definizioni dei profili duplicate all'interno dello stesso file.
- Unisce le impostazioni delle definizioni dei profili duplicate nei file di configurazione e delle credenziali.
- NON unisce le impostazioni se entrambe `[profile foo]` si `[foo]` trovano nello stesso file.
- Utilizza le impostazioni `[profile foo]` se entrambe `[profile foo]` le impostazioni si `[foo]` trovano nel file di configurazione.



- Utilizza il valore dell'ultima impostazione duplicata nello stesso file e profilo.
- Riconosce entrambi ; e # per definire un commento.
- Riconosce ; e # nelle definizioni del profilo definisce un commento, anche se i caratteri sono adiacenti alla parentesi di chiusura.
- Riconosce ; e # definisce un commento solo nell'impostazione dei valori solo se sono preceduti da spazi bianchi.
- Riconosce ; # e tutti i contenuti seguenti nell'impostazione dei valori se non sono preceduti da spazi bianchi.
- Considera le credenziali basate sui ruoli le credenziali con la massima priorità. L'SDK 2.x utilizza sempre credenziali basate sui ruoli se l'utente specifica la proprietà. `role_arn`
- Considera le credenziali basate sulla sessione come credenziali. `second-highest-priority` L'SDK 2.x utilizza sempre credenziali basate sulla sessione se non sono state utilizzate credenziali basate sui ruoli e l'utente specifica le proprietà `aws_access_key_id` `aws_session_token`
- Utilizza le credenziali di base se non vengono utilizzate credenziali basate sul ruolo e sulla sessione e l'utente ha specificato la proprietà. `aws_access_key_id`

## Variabili di ambiente e modifiche alle proprietà del sistema

1.x Variabile di ambiente	1.x Proprietà del sistema	Variabile d'ambiente 2.x	Proprietà di sistema 2.x
AWS_ACCESS_KEY_ID AWS_ACCESS_KEY	<code>aws.accessKeyId</code>	AWS_ACCESS_KEY_ID	<code>aws.accessKeyId</code>
AWS_SECRET_KEY AWS_SECRET_ACCESS_KEY	<code>aws.secretKey</code>	AWS_SECRET_ACCESS_KEY	<code>aws.secretAccessKey</code>
AWS_SESSION_TOKEN	<code>aws.sessionToken</code>	AWS_SESSION_TOKEN	<code>aws.sessionToken</code>
AWS_REGION	<code>aws.region</code>	AWS_REGION	<code>aws.region</code>
AWS_CONFIG_FILE		AWS_CONFIG_FILE	<code>aws.configFile</code>

1.x Variabile di ambiente	1.x Proprietà del sistema	Variabile d'ambiente 2.x	Proprietà di sistema 2.x
AWS_CREDENTIALS_PROFILE_FILE		AWS_SHARED_CREDENTIALS_FILE	<code>aws.sharedCredentialsFile</code>
AWS_PROFILE	<code>aws.profile</code>	AWS_PROFILE	<code>aws.profile</code>
AWS_EC2_METADATA_DISABLED	<code>com.amazonaws.sdk.disableEc2Metadata</code>	AWS_EC2_METADATA_DISABLED	<code>aws.disableEc2Metadata</code>
	<code>com.amazonaws.sdk.ec2MetadataServiceEndpointOverride</code>	AWS_EC2_METADATA_SERVICE_ENDPOINT	<code>aws.ec2MetadataServiceEndpoint</code>
AWS_CONTAINER_CREDENTIALS_RELATIVE_URI		AWS_CONTAINER_CREDENTIALS_RELATIVE_URI	<code>aws.containerCredentialsPath</code>
AWS_CONTAINER_CREDENTIALS_FULL_URI		AWS_CONTAINER_CREDENTIALS_FULL_URI	<code>aws.containerCredentialsFullUri</code>
AWS_CONTAINER_AUTHORIZATION_TOKEN		AWS_CONTAINER_AUTHORIZATION_TOKEN	<code>aws.containerAuthorizationToken</code>

1.x Variabile di ambiente	1.x Proprietà del sistema	Variabile d'ambiente 2.x	Proprietà di sistema 2.x
AWS_CBOR_DISABLED	com.amazonaws.sdk.disableCbor	CBOR_ENABLED	aws.cborEnabled
AWS_ION_BINARY_DISABLE	com.amazonaws.sdk.disableIonBinary	BINARY_ION_ENABLED	aws.binaryIonEnabled
AWS_EXECUTION_ENV		AWS_EXECUTION_ENV	aws.executionEnvironment
	com.amazonaws.sdk.disableCertificateChecking	Non supportato ( <a href="#">funzionalità di richiesta</a> )	Non supportato ( <a href="#">funzionalità di richiesta</a> )
	com.amazonaws.sdk.enableDefaultMetrics	<a href="#">Non supportata</a>	<a href="#">Non supportato</a>
	com.amazonaws.sdk.enableThrottledRetry	<a href="#">Non supportato</a>	<a href="#">Non supportato</a>
	com.amazonaws.regions.RegionUtils.fileOverride	Non supportato ( <a href="#">funzionalità di richiesta</a> )	Non supportato ( <a href="#">funzionalità di richiesta</a> )

1.x Variabile di ambiente	1.x Proprietà del sistema	Variabile d'ambiente 2.x	Proprietà di sistema 2.x
	<code>com.amazonaws.regions.RegionUtils.disableRemote</code>	Non supportato ( <a href="#">funzionalità di richiesta</a> )	Non supportato ( <a href="#">funzionalità di richiesta</a> )
	<code>com.amazonaws.services.s3.disableImplicitGlobalClients</code>	Non supportato ( <a href="#">funzionalità di richiesta</a> )	Non supportato ( <a href="#">funzionalità di richiesta</a> )
	<code>com.amazonaws.sdk.enableInRegionOptimizedMode</code>	Non supportato ( <a href="#">funzionalità di richiesta</a> )	Non supportato ( <a href="#">funzionalità di richiesta</a> )

## Modifiche in Waiters dalla versione 1 alla versione 2

Questo argomento descrive in dettaglio le modifiche alla funzionalità di Waiters dalla versione 1 (v1) alla versione 2 (v2).

Le tabelle seguenti mostrano la differenza in particolare per i camerieri di DynamoDB. I camerieri di altri servizi seguono lo stesso schema.

### Modifiche di alto livello

Le lezioni per camerieri si svolgono nello stesso oggetto Maven del servizio.

Modifica	v1	v2
Dipendenze da Maven	<code>&lt;dependencyManagement&gt;</code>	<code>&lt;dependencyManagement&gt;</code>

Modifica	v1	v2
	<pre> &lt;dependencies&gt;   &lt;dependency&gt;     &lt;groupId&gt; com.amazonaws&lt;/gro upId&gt;     &lt;artifact Id&gt;aws-java-sdk-bom&lt;/ artifactId&gt;     &lt;version&gt; 1.12.680<sup>1</sup>&lt;/version&gt;     &lt;type&gt;pom&lt;/ type&gt;     &lt;scope&gt;im port&lt;/scope&gt;   &lt;/dependency&gt; &lt;/dependencies&gt; &lt;/dependencyManageme nt&gt; &lt;dependencies&gt;   &lt;dependency&gt;     &lt;groupId&gt; com.amazonaws&lt;/gro upId&gt;     &lt;artifact Id&gt;dynamodb&lt;/artif actId&gt;   &lt;/dependency&gt; &lt;/dependencies&gt; </pre>	<pre> &lt;dependencies&gt;   &lt;dependency&gt;     &lt;groupId&gt; software.amazon.aw ssdk&lt;/groupId&gt;     &lt;artifact Id&gt;bom&lt;/artifactId&gt;     &lt;version&gt; 2.25.10<sup>2</sup>&lt;/version&gt;     &lt;type&gt;pom&lt;/ type&gt;     &lt;scope&gt;im port&lt;/scope&gt;   &lt;/dependency&gt; &lt;/dependencies&gt; &lt;/dependencyManageme nt&gt; &lt;dependencies&gt;   &lt;dependency&gt;     &lt;groupId&gt; software.amazon.aw ssdk&lt;/groupId&gt;     &lt;artifact Id&gt;dynamodb&lt;/artif actId&gt;   &lt;/dependency&gt; &lt;/dependencies&gt; </pre>
Nome pacchetto	com.amazonaws.services.dynamodbv2.waiters	software.amazon.awssdk.services.dynamodb.waiters
Nomi delle classi	<a href="#">AmazonDynamoDBWaiters</a>	<ul style="list-style-type: none"> <li>• Sincrono: <a href="#">DynamoDbWaiter</a></li> <li>• Asincrono: <a href="#">DynamoDbAsyncWaiter</a></li> </ul>

<sup>1</sup> Ultima versione. <sup>2</sup> [Ultima versione.](#)

## Modifiche alle API

Modifica	v1	v2
Crea un cameriere	<pre>AmazonDynamoDB client = AmazonDynamoDBClientBuilder      .standard().build( ); AmazonDynamoDBWaiters waiter = client.waiters();</pre>	<p><b>Sincrono:</b></p> <pre>DynamoDbClient client = DynamoDbClient.create(); DynamoDbWaiter waiter = client.waiter();</pre> <p><b>Asincrono:</b></p> <pre>DynamoDbAsyncClient asyncClient =     DynamoDbAsyncClient.create(); DynamoDbAsyncWaiter waiter = asyncClient.waiter();</pre>
Attendi che esista una tabella	<p><b>Sincrono:</b></p> <pre>waiter.tableExists()     .run(new WaiterParameters&lt;&gt;(         new DescribeTableRequest(tableName)));</pre> <p><b>Asincrono:</b></p> <pre>waiter.tableExists()     .runAsync(new WaiterParameters()         .withRequest(new DescribeTableRequest(tableName)),</pre>	<p><b>Sincrono:</b></p> <pre>WaiterResponse&lt;DescribeTableResponse&gt; waiterResponse =     waiter.waitForTableExists(         r -&gt; r.tableName("myTable")); waiterResponse.matched().response()     .ifPresent(System.out::println);</pre> <p><b>Asincrono:</b></p>

Modifica	v1	v2
	<pre> new WaiterHan dler() {     @Override     public     void onWaitSuccess(         AmazonWebServiceRe quest amazonWeb ServiceRequest) {         System.out.println ("Table creation succeeded");     }     @Override     public     void onWaitFai lure(Exception e) {         e.printStackTrace();     } }).get(); </pre>	<pre> waiter.waitUntilTa bleExists(r -&gt;     r.tableName(tableN ame))         .whenComp lete((r, t) -&gt; {             if (t != null) {                 t.printStackTrace();             } else {                 System.out.println(                     "Table creation succeeded");             }         }).join(); </pre>

## Modifiche di configurazione

Modifica	v1	v2
Strategia di sondaggio (numero massimo di tentativi e ritardo fisso)	<pre> MaxAttemptsRetrySt rategy maxAttemp tsRetryStrategy =     new MaxAttemp tsRetryStrategy(10); FixedDelayStrategy fixedDelayStrategy =     new FixedDela yStrategy(3); </pre>	<pre> FixedDelayBackoff Strategy fixedDela yBackoffStrategy =     FixedDela yBackoffStrategy         .create(D uration.ofSeconds( 3)); </pre>

Modifica	v1	v2
	<pre> PollingStrategy     pollingStrategy =         new PollingStrategy(maxAttempts RetryStrategy,      fixedDelayStrategy);  waiter.tableExists().run(     new WaiterParameters&lt;&gt;(         new         DescribeTableRequest(tableName),          pollingStrategy); </pre>	<pre> waiter.waitUntilTableExists(r -&gt; r.tableName(tableName),     c -&gt; c.maxAttempts(10)      .backoffStrategy(fixedDelayBackoffStrategy)); </pre>

## Modifiche in Amazon S3 Transfer Manager dalla versione 1 alla versione 2

Questo argomento descrive in dettaglio le modifiche in Amazon S3 Transfer Manager dalla versione 1 (v1) alla versione 2 (v2).

### Modifiche di alto livello

Modifica	v1	v2
Dipendenze da Maven	<pre> &lt;dependencyManagement&gt;   &lt;dependencies&gt;     &lt;dependency&gt;       &lt;groupId&gt; com.amazonaws&lt;/groupId&gt;       &lt;artifactId&gt;aws-java-sdk-bom&lt;/artifactId&gt;       &lt;version&gt; 1.12.587&lt;/version&gt; </pre>	<pre> &lt;dependencyManagement&gt;   &lt;dependencies&gt;     &lt;dependency&gt;       &lt;groupId&gt; software.amazon.awssdk&lt;/groupId&gt;       &lt;artifactId&gt;bom&lt;/artifactId&gt;       &lt;version&gt; 2.21.21&lt;/version&gt;       &lt;type&gt;pom&lt;/type&gt; </pre>



Modifica	v1	v2
	<pre> &lt;type&gt;pom&lt;/ type&gt; &lt;scope&gt;im port&lt;/scope&gt; &lt;/dependency&gt; &lt;/dependencies&gt; &lt;/dependencyManageme nt&gt; &lt;dependencies&gt; &lt;dependency&gt; &lt;groupId&gt; com.amazonaws&lt;/gro upId&gt; &lt;artifact Id&gt;aws-java-sdk-s3&lt;/ artifactId&gt; &lt;/dependency&gt; &lt;/dependencies&gt; </pre>	<pre> &lt;scope&gt;im port&lt;/scope&gt; &lt;/dependency&gt; &lt;/dependencies&gt; &lt;/dependencyManageme nt&gt; &lt;dependencies&gt; &lt;dependency&gt; &lt;groupId&gt; software.amazon.aw ssdk&lt;/groupId&gt; &lt;artifactId&gt;s3- transfer-manager&lt;/art ifactId&gt; &lt;/dependency&gt; &lt;dependency&gt; &lt;groupId&gt; software.amazon.aw ssdk.crt&lt;/groupId&gt; &lt;artifa ctId&gt;aws-crt&lt;/artifa ctId&gt; &lt;version&gt; 0.28.7<sup>3</sup>&lt;/version&gt; &lt;/dependency&gt; &lt;/dependencies&gt; </pre>
Nome pacchetto	com.amazonaws.serv ices.s3.transfer	software.amazon.aw ssdk.transfer.s3
Class name (Nome classe)	<a href="#">TransferManager</a>	<a href="#">S3TransferManager</a>

1 Ultima versione. <sup>2</sup> [Ultima versione.](#) <sup>3</sup> [Ultima versione.](#)

## Modifiche all'API di configurazione

Impostazione	v1	v2
(procurati un builder)	<pre>TransferManagerBuilder tmBuilder =     TransferManagerBui lder.standard();</pre>	<pre>S3TransferManager. Builder tmBuilder =     S3TransferManager. builder();</pre>
Client S3	<pre>tmBuilder.withS3Cl ient(...); tmBuilder.setS3C lient(...);</pre>	<pre>tmBuilder.s3Client (...);</pre>
Esecutore	<pre>tmBuilder.withExec utorFactory(...); tmBuilder.setExecu torFactory(...);</pre>	<pre>tmBuilder.executor (...);</pre>
Chiudi i pool di thread	<pre>tmBuilder.withShut DownThreadPools(...); tmBuilder.setS hutdownThreadPools (...);</pre>	Non supportato. L'executo r fornito non verrà spento quando S3 viene chiuso TransferManager
Dimensione minima della parte da caricare	<pre>tmBuilder.withMini mumUploadPartSize( ...); tmBuilder.setMinimumU ploadPartSize(...);</pre>	<pre>S3AsyncClient s3 =     S3AsyncClient.crtB uilder().         minimumPa rtSizeInBytes(...)         .build();  tmBuilder.s3Clie nt(s3);</pre>
Soglia di caricamento in più parti	<pre>tmBuilder.withMini mumUploadPartSize( ...);</pre>	<pre>S3AsyncClient s3 =     S3AsyncClient.crtB uilder().</pre>

Impostazione	v1	v2
	<pre>tmBuilder.setMinimumUploadPartSize(...);</pre>	<pre>threshold InBytes(...).build();  tmBuilder.s3Client(s3) ;</pre>
Dimensione minima della parte di copia	<pre>tmBuilder.withMinimumUploadPartSize(...); tmBuilder.setMinimumUploadPartSize(...);</pre>	<pre>S3AsyncClient s3 =     S3AsyncClient.crtBuilder()         .minimumPartSizeInBytes(...)         .build();  tmBuilder.s3Client(s3);</pre>
Soglia di copia in più parti	<pre>tmBuilder.withMinimumUploadPartSize(...); tmBuilder.setMinimumUploadPartSize(...);</pre>	<pre>S3AsyncClient s3 =     S3AsyncClient.crtBuilder()         .threshold         InBytes(...).build();  tmBuilder.s3Client(s3) ;</pre>
Disattiva i download paralleli	<pre>tmBuilder.withDisableParallelDownloads(...); tmBuilder.setDisableParallelDownloads(...);</pre>	<p>Disabilita i download paralleli passando un client S3 standard basato su Java al gestore di trasferimento.</p> <pre>S3AsyncClient s3 =     S3AsyncClient.builder().build();  tmBuilder.s3Client(s3);</pre>

Impostazione	v1	v2
Calcola sempre md5 multipart	<pre>tmBuilder.withAlwaysCalculateMultipartMd5(...); tmBuilder.setAlwaysCalculateMultipartMd5(...);</pre>	Non supportato.

## Modifiche al comportamento



Il trasferimento parallelo richiede un client S3 AWS basato su CRT



[Nell'SDK for Java 2.x, la funzionalità di trasferimento parallelo automatico \(caricamento/download in più parti\) è disponibile tramite il client S3 basato su CRT.AWS](#) Per abilitare la funzionalità di trasferimento parallelo, è necessario aggiungere esplicitamente la dipendenza della [libreria AWS Common Runtime \(CRT\)](#) per massimizzare le prestazioni.

Il solo client S3 AWS basato su CRT, senza utilizzo, offre prestazioni massimizzate dei trasferimenti `S3TransferManager` paralleli. `S3TransferManagerV2` fornisce API aggiuntive che semplificano il trasferimento di file e directory.

La capacità di `S3TransferManager` eseguire trasferimenti paralleli dipende dalla modalità `S3TransferManager` di avvio e dal fatto che la libreria AWS Common Runtime (CRT) sia stata dichiarata come dipendenza.

La tabella seguente descrive tre scenari di inizializzazione per una versione `S3TransferManager v2` con e senza il AWS CRT dichiarato come dipendenza.

Approccio di inizializzazione S3 v2 TransferManager	AWS CRT è dichiarato come dipendenza?	
	sì	no
Inizializza il <b>S3TransferManager</b> senza passare un'istanza <b>S3AsyncClient</b>  Metodo di creazione statico:		

Approccio di inizializzazione S3 v2 TransferManager	AWS CRT è dichiarato o come dipendenza?	
<pre>S3TransferManager.create();</pre> <p>- O -</p> <p>Metodo Builder:</p> <pre>S3TransferManager.builder().build();</pre>	trasferimento parallelo automatico o abilitato	trasferimento parallelo automatico disabilitato
<p>Passa un'<b>S3AsyncClient</b> istanza creata con uno dei metodi del generatore crt* ()</p> <pre>S3AsyncClient s3AsyncClient = S3AsyncClient.crtBuilder().build(); S3TransferManager.builder().s3AsyncClient(s3AsyncClient).build();</pre> <p>- O -</p> <pre>S3AsyncClient s3AsyncClient = S3AsyncClient.crtCreate(); S3TransferManager.builder().s3AsyncClient(s3AsyncClient).build();</pre>	 trasferimento parallelo automatico o abilitato	 errore di runtime

Approccio di inizializzazione S3 v2 TransferManager	AWS CRT è dichiarato come dipendenza?	
<p>Passa un'<b>S3AsyncClient</b> istanza creata con uno dei metodi di creazione standard in modo che il gestore dei trasferimenti non abbia alcun riferimento al CRT</p> <pre data-bbox="126 443 1019 642">S3AsyncClient s3AsyncClient = S3AsyncClient.builder().build(); S3TransferManager.builder().s3AsyncClient(s3AsyncClient).build();</pre> <p>- O -</p> <pre data-bbox="126 751 1019 951">S3AsyncClient s3AsyncClient = S3AsyncClient.create(); S3TransferManager.builder().s3AsyncClient(s3AsyncClient).build();</pre>	<div data-bbox="1094 279 1222 390" style="text-align: center;"></div> <p data-bbox="1062 443 1255 617" style="text-align: center;">trasferimento parallelo automatico disabilitato</p>	<div data-bbox="1338 279 1466 390" style="text-align: center;"></div> <p data-bbox="1305 443 1498 617" style="text-align: center;">trasferimento parallelo automatico disabilitato</p>

## Download parallelo tramite recuperi a intervalli di byte

Quando la funzione di trasferimento parallelo automatico è abilitata, S3 Transfer Manager v2 utilizza recuperi a [intervalli di byte per recuperare parti specifiche](#) dell'oggetto in parallelo (download in più parti). Il modo in cui un oggetto viene scaricato con v2 non dipende da come l'oggetto è stato originariamente caricato. Tutti i download possono trarre vantaggio da un throughput e da una concorrenza elevati.

Al contrario, con S3 Transfer Manager v1, è importante il modo in cui l'oggetto è stato originariamente caricato. S3 Transfer Manager v1 recupera le parti dell'oggetto nello stesso modo in cui le parti sono state caricate. Se un oggetto è stato originariamente caricato come oggetto singolo, S3 Transfer Manager v1 non è in grado di accelerare il processo di download utilizzando richieste secondarie.

## Comportamento di errore

Con S3 Transfer Manager v1, una richiesta di trasferimento di directory fallisce se una delle sottorichieste fallisce. A differenza della versione 1, le future restituite da S3 Transfer Manager v2 vengono completate correttamente anche se alcune richieste secondarie falliscono.

Di conseguenza, è necessario verificare la presenza di errori nella risposta utilizzando il [CompletedDirectoryDownload.failedTransfers\(\)](#) metodo o il [CompletedDirectoryUpload.failedTransfers\(\)](#) metodo anche quando il futuro viene completato correttamente.

## Modifiche nell'utilità per i metadati EC2 dalla versione 1 alla versione 2

Questo argomento descrive in dettaglio le modifiche all'utilità di metadati SDK for Java Amazon Elastic Compute Cloud (EC2) dalla versione 1 (v1) alla versione 2 (v2).

### Modifiche di alto livello

Modifica	v1	v2
Dipendenze da Maven	<pre>&lt;dependencyManagement&gt;   &lt;dependencies&gt;     &lt;dependency&gt;       &lt;groupId&gt; com.amazonaws&lt;/gro upId&gt;       &lt;artifact Id&gt;aws-java-sdk-bom&lt;/ artifactId&gt;       &lt;version&gt; 1.12.587<sup>1</sup>&lt;/version&gt;       &lt;type&gt;pom&lt;/ type&gt;       &lt;scope&gt;im port&lt;/scope&gt;     &lt;/dependency&gt;   &lt;/dependencies&gt; &lt;/dependencyManageme nt&gt; &lt;dependencies&gt;   &lt;dependency&gt;     &lt;groupId&gt; com.amazonaws&lt;/gro upId&gt;     &lt;artifact Id&gt;aws-java-sdk-co re&lt;/artifactId&gt;   &lt;/dependency&gt;</pre>	<pre>&lt;dependencyManagement&gt;   &lt;dependencies&gt;     &lt;dependency&gt;       &lt;groupId&gt; software.amazon.aw ssdk&lt;/groupId&gt;       &lt;artifact Id&gt;bom&lt;/artifactId&gt;       &lt;version&gt; 2.21.21<sup>2</sup>&lt;/version&gt;       &lt;type&gt;pom&lt;/ type&gt;       &lt;scope&gt;im port&lt;/scope&gt;     &lt;/dependency&gt;   &lt;/dependencies&gt; &lt;/dependencyManageme nt&gt; &lt;dependencies&gt;   &lt;dependency&gt;     &lt;groupId&gt; software.amazon.aw ssdk&lt;/groupId&gt;     &lt;artifact Id&gt;imds&lt;/artifactId&gt;   &lt;/dependency&gt;   &lt;dependency&gt;</pre>

Modifica	v1	v2
	<pre>&lt;/dependencies&gt;</pre>	<pre>&lt;groupId&gt; software.amazon.aw ssdk&lt;/groupId&gt; &lt;artifact Id&gt;apache-client 3&lt;/ artifactId&gt; &lt;/dependency&gt; &lt;/dependencies&gt;</pre>
Nome pacchetto	com.amazonaws.util	software.amazon.awssdk.imds
Approccio all'istanziamento	<p>Utilizza metodi di utilità statici; nessuna istanziamento:</p> <pre>String localHostName =     EC2Metada taUtils.getLocalHo stName();</pre>	<p>Usa un metodo di fabbrica statico:</p> <pre>Ec2MetadataClient client = Ec2Metada taClient.create();</pre> <p>Oppure usa un approccio da costruttore:</p> <pre>Ec2MetadataClient client = Ec2Metada taClient.builder()     .endpointMode(Endp ointMode.IPV6)     .build();</pre>
Tipi di clienti	Metodi di utilità solo sincroni: EC2MetadataUtils	<p>Sincrono: Ec2MetadataClient</p> <p>Asincrono: Ec2MetadataAsyncClient</p>

1 Ultima versione. 2 [Ultima versione.](#)



<sup>3</sup> Notate la dichiarazione del `apache-client` modulo per la v2. La versione 2 dell'utilità per i metadati EC2 richiede l'implementazione dell'interfaccia per il client di metadati sincrono o dell'`SdkHttpClient` interfaccia per il client di metadati asincrono. `SdkAsyncHttpClient` La [???](#) sezione mostra l'elenco dei client HTTP che è possibile utilizzare.

## Richiesta di metadati

Nella v1, si utilizzano metodi statici che non accettano parametri per richiedere i metadati per una risorsa EC2. Al contrario, è necessario specificare il percorso della risorsa EC2 come parametro nella v2. La tabella seguente mostra i diversi approcci.

v1	v2
<pre>String userMetaData = EC2MetadataUtils.getUserData();</pre>	<pre>Ec2MetadataClient client = Ec2MetadataClient.create(); Ec2MetadataResponse response =     client.get("/latest/user-data"); String userMetaData =     response.asString();</pre>

Fate riferimento alle [categorie di metadati delle istanze](#) per trovare il percorso da fornire per richiedere una parte di metadati.

### Note

Quando utilizzi un client di metadati di istanza nella v2, dovresti mirare a utilizzare lo stesso client per tutte le richieste di recupero dei metadati.

## Modifiche al comportamento

### Dati JSON

In EC2, l'Instance Metadata Service (IMDS) in esecuzione locale restituisce alcuni metadati come stringhe in formato JSON. [Uno di questi esempi sono i metadati dinamici di un documento di identità di istanza.](#)

L'API v1 contiene metodi separati per ogni parte dei metadati di identità dell'istanza, mentre l'API v2 restituisce direttamente la stringa JSON. Per lavorare con la stringa JSON, puoi utilizzare l'[API Document](#) per analizzare la risposta e navigare nella struttura JSON.

La tabella seguente confronta il modo in cui recuperi i metadati di un documento di identità di istanza in v1 e v2.

Caso d'uso	v1	v2
Recupera la regione	<pre>InstanceInfo instanceInfo =     EC2MetadataUtils.getInstanceInfo(); String region =     instanceInfo.getRegion();</pre>	<pre>Ec2MetadataResponse response =     client.get("/latest/dynamic/instance-identity/document"); Document instanceInfo = response.asDocument(); String region =     instanceInfo.asMap().get("region").asString();</pre>
Recupera l'id dell'istanza	<pre>InstanceInfo instanceInfo =     EC2MetadataUtils.getInstanceInfo(); String instanceId =     instanceInfo.getInstanceId();</pre>	<pre>Ec2MetadataResponse response =     client.get("/latest/dynamic/instance-identity/document"); Document instanceInfo = response.asDocument(); String instanceId =     instanceInfo.asMap().get("instanceId").asString();</pre>
Recupera il tipo di istanza	<pre>InstanceInfo instanceInfo =     EC2MetadataUtils.getInstanceInfo();</pre>	<pre>Ec2MetadataResponse response =     client.get("/latest/dynamic/instance-identity/document");</pre>

Caso d'uso	v1	v2
	<pre>String instanceType = instanceInfo.insta nceType();</pre>	<pre>Document instanceInfo = response.asDocumen t(); String instanceType = instanceInfo.asMap ().get("instanceTy pe").asString();</pre>

## Differenze di risoluzione degli endpoint

La tabella seguente mostra le posizioni controllate dall'SDK per risolvere l'endpoint in IMDS. Le posizioni sono elencate con priorità decrescente.

v1	v2
Proprietà del sistema: <code>com.amazonaws.sdk.ec2MetadataServiceEndpointOverride</code>	Metodo di configurazione di Client Builder: <code>endpoint(...)</code>
Variabile d'ambiente: <code>AWS_EC2_METADATA_SERVICE_ENDPOINT</code>	Proprietà del sistema: <code>aws.ec2MetadataServiceEndpoint</code>
Valore predefinito: <code>http://169.254.169.254</code>	File di configurazione: <code>~.aws/config</code> con l'impostazione <code>ec2_metadata_service_endpoint</code>
	Valore associato a risolto <code>endpoint-mode</code>
	Valore predefinito: <code>http://169.254.169.254</code>

## Risoluzione dell'endpoint in v2

Quando imposti esplicitamente un endpoint utilizzando il builder, quel valore dell'endpoint ha la priorità su tutte le altre impostazioni. Quando viene eseguito il codice seguente, la proprietà di `aws.ec2MetadataServiceEndpoint` sistema e

l'ec2\_metadata\_service\_endpoint impostazione del file di configurazione vengono ignorate, se esistono.

```
Ec2MetadataClient client = Ec2MetadataClient
    .builder()
    .endpoint(URI.create("endpoint.to.use"))
    .build();
```

## modalità Endpoint

Con la v2, è possibile specificare una modalità endpoint per configurare il client di metadati in modo che utilizzi i valori di endpoint predefiniti per IPv4 o IPv6. La modalità endpoint non è disponibile per la versione 1. Il valore predefinito utilizzato per IPv4 è `http://169.254.169.254` e per IPv6 `http://[fd00:ec2::254]`.

La tabella seguente mostra i diversi modi in cui è possibile impostare la modalità endpoint in ordine di priorità decrescente.

		Valori possibili
Metodo di configurazione di Client Builder: <code>endpointMode(...)</code>	<pre>Ec2MetadataClient client = Ec2MetadataClient     .builder()     .endpointMode(EndpointMode.IPV4)     .build();</pre>	<code>EndpointMode.IPV4</code> , <code>EndpointMode.IPV6</code>
Proprietà del sistema	<code>aws.ec2MetadataServiceEndpointMode</code>	IPv4, IPv6 (il caso non ha importanza)
File di configurazione: <code>~.aws/config</code>	<code>ec2_metadata_service_endpoint</code> Impostazione	IPv4, IPv6 (il caso non ha importanza)
Non specificato nei modi precedenti	Viene utilizzato IPv4	

## Come si risolve l'SDK o nella v2 **endpointendpoint-mode**

1. L'SDK utilizza il valore impostato nel codice nel client builder e ignora qualsiasi impostazione esterna. Poiché l'SDK genera un'eccezione se entrambi `endpoint` e `endpointMode` vengono chiamati sul client builder, l'SDK utilizza il valore dell'endpoint indipendentemente dal metodo utilizzato.
2. Se non imposti un valore nel codice, l'SDK esamina la configurazione esterna, prima per le proprietà del sistema e poi per un'impostazione nel file di configurazione.
  - a. L'SDK verifica innanzitutto il valore di un endpoint. Se viene trovato un valore, viene utilizzato.
  - b. Se l'SDK non ha ancora trovato un valore, l'SDK cerca le impostazioni della modalità endpoint.
3. Infine, se l'SDK non trova impostazioni esterne e non hai configurato il client di metadati nel codice, l'SDK utilizza il valore IPv4 di `http://169.254.169.254`

## IMDSv2

Amazon EC2 definisce due approcci per accedere ai metadati delle istanze:

- Instance Metadata Service versione 1 (IMDSv1): approccio di richiesta/risposta
- Instance Metadata Service versione 2 (IMDSv2): approccio orientato alla sessione

La tabella seguente confronta il modo in cui gli SDK Java funzionano con IMDS.

v1	v2
IMDSv2 viene utilizzato per impostazione predefinita	Usa sempre IMDSv2
Tenta di recuperare un token di sessione per ogni richiesta e torna a IMDSv1 se non riesce a recuperare un token di sessione	Mantiene un token di sessione in una cache interna che viene riutilizzata per più richieste

L'SDK for Java 2.x supporta solo IMDSv2 e non ricorre a IMDSv1.

## Differenze di configurazione

La tabella seguente elenca le diverse opzioni di configurazione.

Configurazione	v1	v2
Tentativi	Configurazione non disponibile	Configurabile tramite il metodo builder <code>retryPolicy(...)</code>
HTTP	Timeout di connessione configurabile tramite la variabile di ambiente. <code>AWS_METADATA_SERVICE_TIMEOUT</code> L'impostazione predefinita è 1 secondo.	Configurazione disponibile passando un client HTTP al metodo <code>httpClient(...)</code> builder. Il timeout di connessione predefinito per i client HTTP è di 2 secondi.

### Esempio di configurazione HTTP v2

L'esempio seguente mostra come configurare il client di metadati. Questo esempio configura il timeout della connessione e utilizza il client HTTP Apache.

```

SdkHttpClient httpClient = ApacheHttpClient.builder()
    .connectionTimeout(Duration.ofSeconds(1))
    .build();

Ec2MetadataClient imdsClient = Ec2MetadataClient.builder()
    .httpClient(httpClient)
    .build();

```

## Modifiche nella CloudFront preassegnazione di Amazon dalla versione 1 alla versione 2

Questo argomento descrive in dettaglio le modifiche in Amazon CloudFront dalla versione 1 (v1) alla versione 2 (v2).

### Modifiche di alto livello

Modifica	v1	v2
Dipendenze da Maven	<pre>&lt;dependencyManagement&gt; &lt;dependencies&gt;</pre>	<pre>&lt;dependencyManagement&gt; &lt;dependencies&gt;</pre>

Modifica	v1	v2
	<pre> &lt;dependency&gt;   &lt;groupId&gt; com.amazonaws&lt;/gro upId&gt;     &lt;artifact Id&gt;aws-java-sdk-bom&lt;/ artifactId&gt;     &lt;version&gt; 1.12.587<sup>1</sup>&lt;/version&gt;     &lt;type&gt;pom&lt;/ type&gt;     &lt;scope&gt;im port&lt;/scope&gt;   &lt;/dependency&gt; &lt;/dependencies&gt; &lt;/dependencyManageme nt&gt; &lt;dependencies&gt;   &lt;dependency&gt;     &lt;groupId&gt; com.amazonaws&lt;/gro upId&gt;     &lt;artifact Id&gt;cloudfront&lt;/art ifactId&gt;     &lt;/dependency&gt;   &lt;/dependencies&gt; </pre>	<pre> &lt;dependency&gt;   &lt;groupId&gt; software.amazon.aw ssdk&lt;/groupId&gt;     &lt;artifact Id&gt;bom&lt;/artifactId&gt;     &lt;version&gt; 2.21.21<sup>2</sup>&lt;/version&gt;     &lt;type&gt;pom&lt;/ type&gt;     &lt;scope&gt;im port&lt;/scope&gt;   &lt;/dependency&gt; &lt;/dependencies&gt; &lt;/dependencyManageme nt&gt; &lt;dependencies&gt;   &lt;dependency&gt;     &lt;groupId&gt; software.amazon.aw ssdk&lt;/groupId&gt;     &lt;artifact Id&gt;cloudfront&lt;/art ifactId&gt;     &lt;/dependency&gt;   &lt;/dependencies&gt; </pre>
Nome pacchetto	com.amazonaws.serv ices.cloudfront	software.amazon.aw ssdk.services.clou dfont
Nomi delle classi	<a href="#">CloudFrontUrlSigner</a>  <a href="#">CloudFrontCookieSigner</a>	<a href="#">CloudFrontUtilities</a>  <a href="#">SignedUrl</a>  <a href="#">CannedSignerRequest</a>  <a href="#">CustomSignerRequest</a>

<sup>1</sup> [Ultima versione.](#) <sup>2</sup> [Ultima versione.](#)

## Modifiche alle API

Comportamento	v1	v2
Crea una richiesta predefinita	Gli argomenti vengono passati direttamente all'API.	<pre>CannedSignerRequest cannedRequest =     CannedSig nerRequest.builder()      .resourceUrl(resou rceUrl)      .privateKey(privat eKey)      .keyPairId(keyPairId)      .expirationDate(ex pirationDate)      .build();</pre>
Crea una richiesta personalizzata	Gli argomenti vengono passati direttamente all'API.	<pre>CustomSignerRequest customRequest =     CustomSig nerRequest.builder()      .resourceUrl(resou rceUrl)      .privateKey(keyFile)      .keyPairId(keyPairId)      .expirationDate(ex pirationDate)      .activeDate(active Date)</pre>



Comportamento	v1	v2
		<pre>.ipRange(ipRange)  .build();</pre>
Genera un URL firmato (predefinito)	<pre>String signedUrl =     CloudFrontUrlSigner.getSignedURLWithCannedPolicy(         resourceUrl,         keyPairId, privateKey, expirationDate);</pre>	<pre>CloudFrontUtilities cloudFrontUtilities =     CloudFrontUtilities.create();  SignedUrl signedUrl =      cloudFrontUtilities.getSignedUrlWithCannedPolicy(cannedRequest);  String url = signedUrl.url();</pre>
Genera un cookie firmato (personalizzato)	<pre>CookiesForCustomPolicy cookies =     CloudFrontCookieSigner.getCookiesForCustomPolicy(         resourceUrl,         privateKey, keyPairId, expirationDate,         activeDate,         ipRange);</pre>	<pre>CloudFrontUtilities cloudFrontUtilities =     CloudFrontUtilities.create();  CookiesForCustomPolicy cookies =     cloudFrontUtilities.getCookiesForCustomPolicy(customRequest);</pre>

## Intestazioni dei cookie rifattorizzate nella v2

In Java v1, Java SDK fornisce le intestazioni dei cookie come `file. Map.Entry<String, String>`

```
Map.Entry<String, String> signatureMap = cookies.getSignature();
String signatureKey = signatureMap.getKey(); // "CloudFront-Signature"
```

```
String signatureValue = signatureMap.getValue(); // "[SIGNATURE_VALUE]"
```

L'SDK Java v2 fornisce l'intera intestazione come singola. String

```
String signatureHeaderValue = cookies.signatureHeaderValue(); // "CloudFront-
Signature=[SIGNATURE_VALUE]"
```

## Modifiche nell'analisi degli URI di Amazon S3 dalla versione 1 alla versione 2

Questo argomento descrive in dettaglio le modifiche nell'analisi degli URI di Amazon S3 dalla versione 1 (v1) alla versione 2 (v2.).

### Modifiche di alto livello

Per iniziare ad analizzare un URI S3 nella v1, si crea un'istanza utilizzando un costruttore.

AmazonS3URI Nella v2 si chiama un'istanza di, per `parseUri()` restituire un. `S3Utilities S3URI`

Modifica	v1	v2
Dipendenze da Maven	<pre>&lt;dependencyManagement&gt;   &lt;dependencies&gt;     &lt;dependency&gt;       &lt;groupId&gt; com.amazonaws&lt;/gro upId&gt;       &lt;artifact Id&gt;aws-java-sdk-bom&lt;/ artifactId&gt;       &lt;version&gt; <b>1.12.587</b><sup>1</sup>&lt;/version&gt;       &lt;type&gt;pom&lt;/ type&gt;       &lt;scope&gt;im port&lt;/scope&gt;     &lt;/dependency&gt;   &lt;/dependencies&gt; &lt;/dependencyManageme nt&gt; &lt;dependencies&gt;   &lt;dependency&gt;</pre>	<pre>&lt;dependencyManagement&gt;   &lt;dependencies&gt;     &lt;dependency&gt;       &lt;groupId&gt; software.amazon.aw ssdk&lt;/groupId&gt;       &lt;artifact Id&gt;bom&lt;/artifactId&gt;       &lt;version&gt; <b>2.21.21</b><sup>2</sup>&lt;/version&gt;       &lt;type&gt;pom&lt;/ type&gt;       &lt;scope&gt;im port&lt;/scope&gt;     &lt;/dependency&gt;   &lt;/dependencies&gt; &lt;/dependencyManageme nt&gt; &lt;dependencies&gt;   &lt;dependency&gt;</pre>

Modifica	v1	v2
	<pre>&lt;groupId&gt; com.amazonaws&lt;/gro upId&gt; &lt;artifact Id&gt;s3&lt;/artifactId&gt; &lt;/dependency&gt; &lt;/dependencies&gt;</pre>	<pre>&lt;groupId&gt; software.amazon.aw ssdk&lt;/groupId&gt; &lt;artifact Id&gt;s3&lt;/artifactId&gt; &lt;/dependency&gt; &lt;/dependencies&gt;</pre>
Nome pacchetto	com.amazonaws.serv ices.s3	software.amazon.aw ssdk.services.s3
Nomi delle classi	<a href="#">AmazonS3URI</a>	<a href="#">S3URI</a>

<sup>1</sup> [Ultima versione.](#) <sup>2</sup> [Ultima versione.](#)

## Modifiche alle API

Comportamento	v1	v2
Analizza un URI S3.	<pre>URI uri = URI.creat e( "https://s3.amazon aws.com");  AmazonS3Uri s3Uri =     new AmazonS3U RI(uri, false);</pre>	<pre>S3Client s3Client =     S3Client.create(); S3Utilities s3Utiliti es =     s3Client.utilities ();  S3Uri s3Uri =     s3Utilities.parseU ri(uri);</pre>
Recupera il nome del bucket da un URI S3.	<pre>String bucket =     s3Uri.getBucket();</pre>	<pre>Optional&lt;String&gt; bucket = s3Uri.bucket();</pre>
Recupera la chiave.	<pre>String key = s3Uri.get Key();</pre>	<pre>Optional&lt;String&gt; key =     s3Uri.key();</pre>

Comportamento	v1	v2
Recupera la regione.	<pre>String region = s3Uri.getRegion();</pre>	<pre>Optional&lt;Region&gt; region = s3Uri.region();  String region; if (s3Uri.region().is Present()) {     region = s3Uri.reg ion().get().id(); }</pre>
Recupera se l'URI S3 è in stile percorso.	<pre>boolean isPathStyle = s3Uri.isPathStyle();</pre>	<pre>boolean isPathStyle = s3Uri.isPathStyle();</pre>
Recupera l'ID della versione.	<pre>String versionId = s3Uri.getVersionId();</pre>	<pre>Optional&lt;String&gt; versionId =     s3Uri.firstMatchin gRawQueryParameter ("versionId");</pre>
Recupera i parametri dell'interrogazione.	N/D	<pre>Map&lt;String, List&lt;Stri ng&gt;&gt; queryParams =     s3Uri.rawQueryPara meters();</pre>

## Modifiche al comportamento

### Codifica URL

v1 offre la possibilità di inserire un flag per specificare se l'URI deve essere codificato come URL. Il valore predefinito è `true`.

Nella v2, la codifica degli URL non è supportata. Se lavori con chiavi oggetto o parametri di query che contengono caratteri riservati o non sicuri, devi codificarli tramite URL. Ad esempio, è necessario sostituire uno spazio bianco " " con. `%20`

## Modifiche nell'API IAM Policy Builder dalla versione 1 alla versione 2

Questo argomento descrive in dettaglio le modifiche all'API IAM Policy Builder dalla versione 1 (v1) alla versione 2 (v2).

### Modifiche di alto livello

Modifica	v1	v2
Dipendenze da Maven	<pre> &lt;dependencyManagement&gt;   &lt;dependencies&gt;     &lt;dependency&gt;       &lt;groupId&gt; com.amazonaws&lt;/gro upId&gt;       &lt;artifact Id&gt;aws-java-sdk-bom&lt;/ artifactId&gt;       &lt;version&gt; 1.12.587&lt;sup&gt;1&lt;/sup&gt;&lt;/version&gt;       &lt;type&gt;pom&lt;/ type&gt;       &lt;scope&gt;im port&lt;/scope&gt;     &lt;/dependency&gt;   &lt;/dependencies&gt; &lt;/dependencyManageme nt&gt; &lt;dependencies&gt;   &lt;dependency&gt;     &lt;groupId&gt; com.amazonaws&lt;/gro upId&gt;     &lt;artifact Id&gt;aws-java-sdk-co re&lt;/artifactId&gt;     &lt;/dependency&gt;   &lt;/dependencies&gt; </pre>	<pre> &lt;dependencyManagement&gt;   &lt;dependencies&gt;     &lt;dependency&gt;       &lt;groupId&gt; software.amazon.aw ssdk&lt;/groupId&gt;       &lt;artifact Id&gt;bom&lt;/artifactId&gt;       &lt;version&gt; 2.21.21&lt;sup&gt;2&lt;/sup&gt;&lt;/version&gt;       &lt;type&gt;pom&lt;/ type&gt;       &lt;scope&gt;im port&lt;/scope&gt;     &lt;/dependency&gt;   &lt;/dependencies&gt; &lt;/dependencyManageme nt&gt; &lt;dependencies&gt;   &lt;dependency&gt;     &lt;groupId&gt; software.amazon.aw ssdk&lt;/groupId&gt;     &lt;artifact Id&gt;iam-policy-buil der&lt;/artifactId&gt;     &lt;/dependency&gt;   &lt;/dependencies&gt; </pre>

Modifica	v1	v2
Nome pacchetto	<code>com.amazonaws.auth.policy</code>	<code>software.amazon.awssdk.policybuilder.iam</code>
Nomi delle classi	<p><a href="#">Policy</a></p> <p><a href="#">Statement</a></p> <ul style="list-style-type: none"> <li>• <a href="#">Dichiarazione. Effetto</a></li> <li>• <a href="#">IdentityManagementActions</a></li> <li>• <a href="#">Resource (Risorsa)</a></li> <li>• <a href="#">Principale</a></li> <li>• <a href="#">Condition</a></li> </ul>	<p><a href="#">IamPolicy</a></p> <p><a href="#">IamStatement</a></p> <ul style="list-style-type: none"> <li>• <a href="#">IamEffect</a></li> <li>• <a href="#">IamAction</a></li> <li>• <a href="#">IamResource</a></li> <li>• <a href="#">IamPrincipal</a></li> <li>• <a href="#">IamCondition</a></li> <li>• <a href="#">IamConditionOperator</a></li> <li>• <a href="#">IamConditionKey</a></li> </ul>

<sup>1</sup> Ultima versione. <sup>2</sup> [Ultima versione.](#)

## Modifiche alle API

Impostazione	v1	v2
Crea un'istanza di una politica	<pre>Policy policy = new Policy();</pre>	<pre>IamPolicy.Builder policyBuilder = IamPolicy.builder(); ... IamPolicy policy = policyBuilder.buil d();</pre>
Imposta id	<pre>policy.withtId(...); policy.setId(...);</pre>	<pre>policyBuilder.id(...);</pre>

Impostazione	v1	v2
Impostare la versione	N/A: utilizza la versione predefinita di 2012-10-17	<pre>policyBuilder.version(...);</pre>
Crea una dichiarazione	<pre>Statement statement =     new Statement       (Effect.Allow)         .withActions(...)         .withConditions(...)         .withId(..)         .withPrincipals(...)         .withResources(...);</pre>	<pre>IamStatement statement =     IamStatement.builder()       .effect(IamEffect.ALLOW)       .actions(...)       .notActions(...)       .conditions(...)       .sid(...)       .principals(...)       .notPrincipals(...)       .resources(...)       .notResources(...)       .build();</pre>
Imposta dichiarazione	<pre>policy.withStatements(statement); policy.setStatements(statement);</pre>	<pre>policyBuilder.addStatement(statement);</pre>

## Differenze nella creazione di una dichiarazione

### Azioni

#### v1

L'SDK v1 dispone [enum di tipi di](#) azioni di servizio che rappresentano [Action](#) elementi di una dichiarazione politica. I seguenti enum tipi sono alcuni esempi.

- [IdentityManagementActions](#)
- [DynamoDBv2Actions](#)
- [SQSActions](#)

L'esempio seguente mostra la `SendMessage` costante per `SQSActions`.

```
Action action = SQSActions.SendMessage;
```

Non è possibile specificare un [NotAction](#) elemento in un'istruzione in v1.

#### v2

Nella v2, l'[IamAction](#) interfaccia rappresenta tutte le azioni. Per specificare un elemento di [azione specifico del servizio](#), passate una stringa al `create` metodo come illustrato nel codice seguente.

```
IamAction action = IamAction.create("sqs:SendMessage");
```

È possibile specificare un'istruzione [NotAction](#) for a con v2, come illustrato nel codice seguente.

```
IamAction action = IamAction.create("sqs:SendMessage");  
IamStatement.builder().addNotAction(action);
```

### Condizioni

#### v1

Per rappresentare le condizioni delle dichiarazioni, l'SDK v1 utilizza sottoclassi di [Condition](#)

- [ArnCondition](#)



- [BooleanCondition](#)
- [DateCondition](#)
- [IpAddressCondition](#)
- [NumericCondition](#)
- [StringCondition](#)

Ogni `Condition` sottoclasse definisce un enum tipo di confronto per aiutare a definire la condizione. Ad esempio, quanto segue mostra un [confronto di stringhe](#) non simile per una condizione.

```
Condition condition = new StringCondition(StringComparisonType.StringNotLike, "key", "value");
```

v2

Nella v2, si crea una condizione per una dichiarazione politica utilizzando [IamCondition](#) e fornendo un [IamConditionOperator](#), che contiene enums per tutti i tipi.

```
IamCondition condition = IamCondition.create(IamConditionOperator.STRING_NOT_LIKE, "key", "value");
```

Risorse

v1

L'[Resource](#) elemento di una dichiarazione politica è rappresentato dalla classe dell'SDK. [Resource](#) L'ARN viene fornito come stringa nel costruttore. Le seguenti sottoclassi forniscono costruttori di convenienza.

- [S3 BucketResource](#)
- [S3 ObjectResource](#)
- [SQS QueueResource](#)

Nella v1, è possibile specificare un [NotResource](#) elemento per a [Resource](#) chiamando il `withIsNotType` metodo come illustrato nella seguente istruzione.

```
Resource resource = new Resource("arn:aws:s3:::mybucket").withIsNotType(true);
```

## v2

Nella v2, si crea un [Resource](#) elemento passando un ARN al `IamResource.create` metodo.

```
IamResource resource = IamResource.create("arn:aws:s3:::mybucket");
```

An [IamResource](#) può essere impostato come [NotResource](#) elemento come mostrato nel frammento seguente.

```
IamResource resource = IamResource.create("arn:aws:s3:::mybucket");  
IamStatement.builder().addNotResource(resource);
```

`IamResource.ALL` rappresenta tutte le risorse.

## Principali

### v1

L'SDK v1 offre le seguenti [Principal](#) classi per rappresentare i tipi di principali che includono tutti i membri:

- `AllUsers`
- `AllServices`
- `AllWebProviders`
- `All`

Non è possibile aggiungere un [NotPrincipal](#) elemento a un'istruzione.

### v2

Nella v2, `IamPrincipal.ALL` rappresenta tutti i principali:

Per rappresentare tutti i membri in altri tipi di principali, usa [IamPrincipalType](#) le classi quando crei un `IamPrincipal`

- `IamPrincipal.create(IamPrincipalType.AWS, "*")` per tutti gli utenti.
- `IamPrincipal.create(IamPrincipalType.SERVICE, "*")` per tutti i servizi.
- `IamPrincipal.create(IamPrincipalType.FEDERATED, "*")` per tutti i provider web.

- `IamPrincipal.create(IamPrincipalType.CANONICAL_USER, "*")` per tutti gli utenti canonici.

È possibile utilizzare il `addNotPrincipal` metodo per rappresentare un [NotPrincipal](#) elemento quando si crea una dichiarazione politica, come illustrato nella seguente dichiarazione.

```
IamPrincipal principal = IamPrincipal.create(IamPrincipalType.AWS,
    "arn:aws:iam::444455556666:root");
IamStatement.builder().addNotPrincipal(principal);
```

## Uso SDK for Java 1.x side-by-side

Nei progetti è possibile utilizzare entrambe le versioni di AWS SDK for Java.

Di seguito viene mostrato un esempio del `pom.xml` file per un progetto che utilizza Amazon S3 dalla versione 1.x e DynamoDB dalla versione 2.16.1.

### Example Esempio di POM

Questo esempio mostra una voce di `pom.xml` file per un progetto che utilizza entrambe le versioni 1.x e 2.x dell'SDK.

```
<dependencyManagement>
  <dependencies>
    <dependency>
      <groupId>com.amazonaws</groupId>
      <artifactId>aws-java-sdk-bom</artifactId>
      <version>1.12.1</version>
      <type>pom</type>
      <scope>import</scope>
    </dependency>
    <dependency>
      <groupId>software.amazon.awssdk</groupId>
      <artifactId>bom</artifactId>
      <version>2.16.1</version>
      <type>pom</type>
      <scope>import</scope>
    </dependency>
  </dependencies>
</dependencyManagement>
```

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

# Chiave OpenPGP per AWS SDK for Java

Tutti gli artefatti Maven disponibili pubblicamente per il AWS SDK for Java sono firmati utilizzando lo standard OpenPGP. La chiave pubblica necessaria per verificare la firma di un artefatto è disponibile nella sezione seguente.

## Chiave attuale

La tabella seguente mostra le informazioni chiave di OpenPGP per le versioni correnti di SDK for Java 1x e SDK for Java 2.x.

ID chiave	0xAC107B386692DADD
Type	RSA
Size	4096/4096
Creato	30-06-2016
Scade	2024-10-08
ID utente	AWSSDK e strumenti <@amazon.com> aws-dr-tools
Impronta digitale chiave	FEB9 209F 2F2F 3F46 6484 1E55 AC10 7B38 6692 PAPA

Per copiare la seguente chiave pubblica OpenPGP per l'SDK for Java negli appunti, seleziona l'icona «Copia» nell'angolo in alto a destra.

```
-----BEGIN PGP PUBLIC KEY BLOCK-----
```

```
xsFNBFd1gAUBEACqbmFbxdJgz1lD7wr1skQA1LLuSAC4p8ny9u/D2zLR8Ynk3Yz
mzJuQ+Kfjne2t+xTDex6MPJlMYp0viSWsX2psgvdmeyUpW9ap01rThNYkc+W5fRc
buFehfbi9LSATZGJi8RG0sCCr5FsYVz0gEk85M2+PeM24cXhQIOZtQUjswX/pdk/
KduGtZASqNAYLKR0mRODzUuaokLPo24pfm9bnr1RnRtw5ktPAA5bM9ZZaGKriej
kT2lPffBbjp8F5AZvmGLtNm2Cmg4FKBvI04SQjy2jjrQ3wBzi5Lc9HTxDuHK/rtV
u6PewUe2WP1nxlXenhMZU1UK4YoSB9E9StQ2VxQiySLHSdxR7Ma4WgYdVLn9b0ie
```

```
nj3QxLuQ1ZUKF79ES6JaM4t0z1gGcQeU1+Uk1gjFLuKwmzWRdEIFfxMyvH6qgKnd
U+DioH5mcUwhwffAAAsuIJyAdMIEUYh7IfzJJXQf+ff+Xf0Cl6by0JFWrIGQkAzMu
CEvaCfwtHC2Lpzo33/WRFEMAuzzd0QJ4uz4xFFvaS0SZHMLHWI9YV/+Pea3X99Ms
0Nlek/LolAJh67MynHeVB0HKrq+fluorWepQivctzN6Y1N0kx5naTPGGaKWK7G2q
TbcY5SMnkIwFLFSougj0Fvmjczq8iZRwYxWA+i+LQvsR9WEXEiQffIWRoQARAQAB
zsFNBFd1gAUBEAC8zNARpWb3dPMThL2xAY+fs60vXdB1Sk0tYJpDWPfGvo0d+VQ+
hV6Xu1GAHAS6xG1WHysPT9KejIRSgLG+e9CaM5yhsxNa1WFGUM4Q9ESo3t+a75Go
7xHIxgFjC046/06Vh3g9N/PREeuG8zkZ3H2v5fmD+ejyPgk4W9sFL00zjRiZD0FK
VYR/j9uenEC/2NBcLuFy3q6cDfmCoDE0062kXMnaGz3knzEK/X1SkcjsxRDq7zaQ
lQ1Kou+3dICwy4x5SjQ8j1+eeeEvF2C2/dXmDohb57tqUwioohMUQkmCtvZgEHjy
pUwgp0MTo25gWxkvJlSJKU0b6b1786WnySIzF2gxq1kkEmB14RAssQkeXjrSmGws
MDyHNqyJeYFus18sPaSpo+V2n0z+2B070Uq+wmf1S5A5FpegH0PZzzoNZo8I6Qxa
Zje9YSZUijGmZIdEBleRVt3Svhi8MY1nasd4bW2RK1sr7plkBf8QRe6biiQRf3KD
0Sn5CbmXpAcHJ1ZHRRdkXZDNQC6vCJxsy1300TrhJtAV1Yq347uyUbVi291ISVg
roUVtprismHoEk5Go0THbg9SCSt+xi/FiJQC+ubWmIGXoFKMR3UmhDnnzobKcbtnbs
/Hd981FdVghYYvq//gTakJk0WxfGq030wtXRndPOA0T+qhP3TE+LtGRJ+wARAQAB
wsF1BBgBCgAPBQJXdYAFaHsMBQkHhh+AAAoJEKwQezhmktrdTyEP/0H0VWHwQsaW
jMrGj000MFzxGUo8SBmYYTBs29VM8wBGDsPkYCjeZzU16i9iqDpDqxyqmTigcjH
V8CDx/6xsMBLG2yKaKZ4m3+Yn0Qf/sQkyCvqiyMF9mS7pDYWy+mPhPuw8TDIfiqg
VhzjSpIMFWPqxVjn6KKbPN/QASr3Pf0cuP6qpHG+NAM6Q5dYkCebyvwzLmg1sVni
l6iSyJd1jBj3D34XrgWS9buyxBB2CjIM76WxfNViJ9zAaPI78X9v6PpDGn0kg6oL
zrusrvBjoZknKQm0SZ+41fx6xvrTPs8uPEzevzJB1kke6kw9+KagY8mrVX1ZenRg
+sY/4vxJreYWQeq167ggx+wFjKDcfhZA7m70LH0DysrGVCLcmuinUBaN1HmLDcGY
XZ+kMCoXf0bpuCVByQmNJgEb47EIFlx/+TEeNHKM0+22xL1atFzXfkEVZck+NghL
ZyFDhS3g1bma7puU7r752uiJjA6Iv8+kHDXi+/V7GNpuiEFUYh69QQ2//CS5H51o
sC/Bkb9evSn/Lp8dMubtWAaXDGJMgw9vqZ55N02NK0fvF/IKHnGkvH28rv00PCv0
WTA/MClv28y0PrSvcmXnduLtkBEX7TISMPW+n+0Ta63/z4YFFEZ7sFLrEm3Q3vJ
MN3mE5i3cw+JGXPSu0nTtgqk/oZv//SS
=Z9u3
-----END PGP PUBLIC KEY BLOCK-----
```

# Cronologia dei documenti

Questo argomento descrive le modifiche importanti apportate alla AWS SDK for Java Developer Guide nel corso della sua storia.

Questa guida è stata pubblicata l'ultima volta il 21 maggio 2024.

Modifica	Descrizione	Data
<a href="#">Come impostare il TTL JVM</a>	Rimuovi le istruzioni per impostare <code>networkaddress.cache.ttl</code> la proprietà di sicurezza utilizzando una proprietà di sistema della riga di comando <code>java</code> .	21 maggio 2024
<a href="#">the section called “Riduci i tempi di avvio dell'SDK per AWS Lambda”</a>	Aggiorna i consigli del client HTTP per ridurre i tempi di avvio per AWS Lambda	14 maggio 2024
<a href="#">the section called “Metriche dei client di assistenza”</a>	Riorganizza gli elementi della tabella delle metriche	1 maggio 2024
<a href="#">the section called “Risoluzione dei problemi”</a>	Aggiungi un argomento per la risoluzione dei problemi.	26 aprile 2024
<a href="#">the section called “Metriche raccolte con ogni richiesta”</a>	Aggiungi nuove metriche riportate dall'SDK.	26 aprile 2024
<a href="#">the section called “Imposta il TTL JVM per le ricerche dei nomi DNS”</a>	Modifica il TTL di ricerca DNS consigliato a 5 secondi.	23 aprile 2024
<a href="#">the section called “Dal nome del pacchetto alle mappature ArtifactID”</a>	Aggiungi il nome del pacchetto all'argomento di mappatura Maven <code>ArtifactID</code> .	17 aprile 2024

Modifica	Descrizione	Data
<a href="#">the section called “Usa le metriche SDK”</a>	Aggiungi i dettagli di configurazione alla sezione delle metriche.	12 aprile 2024
<a href="#">the section called “API IAM Policy Builder”</a>	Aggiungi le informazioni sulla migrazione dell'API IAM Policy Builder.	11 aprile 2024
<a href="#">???</a>	Aggiorna le informazioni sul proxy HTTP.	3 aprile 2024
<a href="#">the section called “In modo sicuro”</a>	Aggiungi istruzioni per disabilitare iMDSv1.	14 marzo 2024
<a href="#">the section called “step-by-step Istruzioni S”</a>	Aggiungi istruzioni per step-by-step la migrazione.	8 marzo 2024
<a href="#">Migrare alla versione 2</a>	Aggiorna argomento sulla migrazione.	14 febbraio 2024
<a href="#">the section called “Configurazione AWS dei client HTTP basati su CRT”</a>	Aggiungi informazioni sul client HTTP sincrono basato su AWS CRT.	5 gennaio 2024
<a href="#">the section called “Amazon Cognito Identity”</a> e <a href="#">the section called “Provider di identità Amazon Cognito”</a>	Gli esempi di Amazon Cognito sono stati spostati nella sezione Code Examples.	28 dicembre 2023
<a href="#">Usa le funzionalità dell'SDK</a>	L'argomento relativo alle funzionalità dell'SDK è stato rielaborato.	11 dicembre 2023
<a href="#">Chiave OpenPGP</a>	Fornisci la chiave OpenPGP corrente.	6 dicembre 2023



Modifica	Descrizione	Data
<a href="#">the section called “Modifiche alla serializzazione”</a>	Descrivi le differenze di serializzazione tra v1 e v2 dell'SDK for Java.	5 dicembre 2023
<a href="#">the section called “S3 Transfer Manager”</a>	Aggiungi una sezione che descrive in dettaglio le modifiche apportate a S3 Transfer Manager dalla versione 1 alla versione 2.	13 novembre 2023
<a href="#">the section called “Riferimento all'annotazione”</a>	Aggiungi un elenco di annotazioni di classi di dati che possono essere utilizzati e con il DynamoDB Enhanced Client.	30 ottobre 2023
<a href="#">???</a>	Aggiungi informazioni sullo stato della migrazione di librerie e utilità da SDK for Java v1.x a v2.x	17 ottobre 2023
<a href="#">???</a>	Aggiorna l'argomento relativo alla configurazione di Gradle	17 ottobre 2023
<a href="#">the section called “Ignora gli attributi nulli degli oggetti nidificati”</a>	Aggiungi informazioni sull'annotazione DynamoDB Enhanced Client. @DynamoDbIgnoreNulls	22 settembre 2023
<a href="#">the section called “Accesso interregionale”</a>	Aggiungi informazioni sull'accesso interregionale ai bucket Amazon S3.	31 agosto 2023

Modifica	Descrizione	Data
<a href="#">the section called “Conserva oggetti vuoti”</a>	Aggiungi una sezione che descrive l'annotazione. <code>@DynamoDbPreserveEmptyObject</code>	25 agosto 2023
<a href="#">???</a>	Aggiorna la sezione client del servizio.	15 agosto 2023
<a href="#">the section called “Consigli per i clienti”</a>	Dalla versione 0.23, AWS CRT supporta sistemi operativi basati su musl come Alpine Linux. Le raccomandazioni dei client HTTP ora riflettono il supporto musl.	11 agosto 2023
<a href="#">the section called “Crea politiche IAM”</a>	Aggiungi la sezione API IAM Policy Builder	31 luglio 2023
<a href="#">the section called “Inizia a usare”</a>	Correggi diversi frammenti nella sezione Get Started dell'argomento DynamoDB Enhanced Client.	24 luglio 2023
<a href="#">the section called “Configurazione dei proxy HTTP”</a>	Aggiungi informazioni ed esempi di supporto per i proxy HTTP per ogni client HTTP.	2 giugno 2023
Riorganizza il sommario	Promuovi le <a href="#">Esempi di codice</a> sezioni e <a href="#">Lavora con Servizi AWS</a> le voci del sommario di primo livello.	24 maggio 2023
<a href="#">the section called “Aggiungi dipendenza dalla registrazione”</a>	Mostra le dipendenze di Gradle nella sezione di registrazione.	23 maggio 2023

Modifica	Descrizione	Data
<a href="#">the section called “Lavora con risultati impaginati”</a>	Aggiorna l'argomento dell'impaginazione.	18 maggio 2023
<a href="#">the section called “Configura un progetto Gradle”</a>	Aggiorna la configurazione del progetto Gradle.	3 maggio 2023
<a href="#">API client avanzata per DynamoDB</a>	È stato rilasciato l'argomento Rewritten DynamoDB Enhanced Client API.	28 aprile 2023
<a href="#">Aggiorna le istruzioni introdotte del tutorial</a>	Archetipo Maven modificato per includere l'opzione per CredentialsProvider; istruzioni modificate di conseguenza.	11 aprile 2023
<a href="#">the section called “Consigli per i clienti”</a>	Aggiungi una guida decisionale al client HTTP	30 marzo 2023
Aggiornamenti delle best practice di IAM	Guida aggiornata per l'allineamento alle best practice IAM. Per ulteriori informazioni, consulta <a href="#">Best practice per la sicurezza in IAM</a> .	14 marzo 2023
<a href="#">the section called “Ricarica le credenziali del profilo”</a>	Aggiungi una sezione sul ricaricamento delle credenziali del profilo.	9 febbraio 2023
<a href="#">the section called “Configurazione AWS dei client HTTP basati su CRT”</a>	Argomento di aggiornamento per la versione GA.	8 febbraio 2023
<a href="#">the section called “Utilizzare i metadati delle istanze Amazon EC2.”</a>	Aggiungi un esempio guidato per il client Java SDK per il servizio di metadati dell'istanza Amazon S3.	1 febbraio 2023

Modifica	Descrizione	Data
<a href="#">the section called “Usa un client S3 performante”</a>	Aggiungi una sezione per il client S3 basato su AWS CRT.	19 dicembre 2022
<a href="#">the section called “Trasferisci file e directory”</a>	Aggiorna gli esempi di Amazon S3 Transfer Manager per la versione GA.	19 dicembre 2022
<a href="#">the section called “Best practice”</a>	È stata aggiunta la sezione sulle migliori pratiche.	18 novembre 2022
<a href="#">the section called “Carica credenziali temporanee da un processo esterno”</a>	È stata aggiunta una sezione sul caricamento delle credenziali da un processo esterno.	15 novembre 2022
<a href="#">the section called “Metriche dei client di assistenza”</a>	Elenco delle metriche aggiornato con i requisiti di utilizzo del client HTTP.	9 novembre 2022
<a href="#">the section called “Trasferisci file e directory”</a>	Codice di esempio corretto.	2 novembre 2022
<a href="#">the section called “Riduci i tempi di avvio dell'SDK per AWS Lambda”</a>	Sezione aggiornata con opzioni aggiuntive per ridurre i tempi di avvio di Lambda.	1 novembre 2022
<a href="#">the section called “Client HTTP”</a>	Sono state aggiunte informazioni di configurazione per coprire tutti i client HTTP nell'SDK.	26 ottobre 2022
<a href="#">the section called “Registrazione”</a>	Argomento di registrazione aggiornato per includere i dettagli della registrazione via cavo per tutti i client HTTP.	4 ottobre 2022

Modifica	Descrizione	Data
<a href="#">the section called “AWS servizi di database”</a>	È stata aggiunta una sezione panoramica dei servizi di AWS database e dell'SDK for Java 2.x.	13 settembre 2022
<a href="#">EC2-Classic Networking viene ritirato</a>	EC2-Classic andrà in pensione il 15 agosto 2022.	28 luglio 2022
<a href="#">the section called “Opzioni di autenticazione aggiuntive”</a>	Aggiornamento alla dipendenza richiesto per l'autenticazione Single Sign-On.	18 luglio 2022
<a href="#">the section called “Transport Layer Security (TLS)”</a>	Aggiorna le informazioni di sicurezza TLS.	8 aprile 2022
<a href="#">the section called “Opzioni di autenticazione aggiuntive”</a>	Sono state aggiunte ulteriori informazioni sulla configurazione e l'utilizzo delle credenziali.	22 febbraio 2021
<a href="#">the section called “Configura un progetto GraalVM Native Image”</a>	Nuovo argomento per la configurazione di un progetto GraalVM Native Image.	18 febbraio 2021
<a href="#">the section called “Sondaggio sugli stati delle risorse”</a>	Waiters è stato rilasciato; è stato aggiunto un argomento per la nuova funzionalità.	30 settembre 2020
<a href="#">the section called “Usa le metriche SDK”</a>	Metriche rilasciate; è stato aggiunto un argomento per la nuova funzionalità.	17 agosto 2020
<a href="#">the section called “Amazon SNS”</a>	Sono stati aggiunti argomenti di esempio per Amazon SNS.	30 maggio 2020

Modifica	Descrizione	Data
<a href="#">the section called “Riduci i tempi di avvio dell'SDK per AWS Lambda”</a>	È stato aggiunto un argomento sulle prestazioni delle AWS Lambda funzioni.	29 maggio 2020
<a href="#">the section called “Imposta il TTL JVM per le ricerche dei nomi DNS”</a>	È stato aggiunto l'argomento di caching DNS JVM TTL.	27 aprile 2020
<a href="#">the section called “Configurare un progetto Apache Maven”, the section called “Configura un progetto Gradle”</a>	Nuovi argomenti di configurazione di Maven e Gradle.	21 aprile 2020
<a href="#">the section called “Transport Layer Security (TLS)”</a>	Aggiunto TLS 1.2 alla sezione sulla sicurezza.	19 marzo 2020
<a href="#">the section called “Sottoscrizione ad Amazon Kinesis Data Streams”</a>	Aggiunti esempi Kinesis di stream.	2 agosto 2018
<a href="#">the section called “Lavora con risultati impaginati”</a>	Aggiunto argomento paginazione automatica.	5 Aprile 2018
<a href="#">???</a>	Sono stati aggiunti argomenti di esempio per IAM Amazon EC2, CloudWatch e DynamoDB.	29 dicembre 2017
<a href="#">the section called “Amazon S3”</a>	È stato aggiunto l'esempio getObject per Amazon S3.	7 agosto 2017
<a href="#">the section called “Usa la programmazione asincrona”</a>	Aggiunto argomento asincrono .	4 agosto 2017
<a href="#">Versione GA della versione 2.x AWS SDK for Java</a>	AWS SDK for Java è stata rilasciata la versione 2 (v2).	28 giugno 2017

Le traduzioni sono generate tramite traduzione automatica. In caso di conflitto tra il contenuto di una traduzione e la versione originale in Inglese, quest'ultima prevarrà.