Guida per gli sviluppatori per la versione 2.x

AWS SDK for Java 2.x



Copyright © 2025 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

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

Copyright © 2025 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

Che cos'è il 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
Nozioni di base 3
Configurazione 4
Panoramica della configurazione 4
Installa i prerequisiti di sviluppo Java 5
Configurare un progetto Apache Maven 6
Configura un progetto Gradle 12
Configurare un progetto GraalVM Native Image 18
Autenticazione con AWS 20
Configurazione per l'autenticazione 20
Opzioni di autenticazione aggiuntive 22
Creazione di un'applicazione semplice 22
Fase 1: Configurazione per questo tutorial 22
Fase 2: Creare il progetto 23
Passaggio 3: scrivere il codice 27
Fase 4: Compilare ed eseguire l'applicazione 32
Passaggi successivi 33
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
Chiudere il client di servizio 36
Effettua richieste 36
Utilizzate le richieste per sovrascrivere la configurazione del client 37
Gestisci le risposte 38
Gestisci le eccezioni 39
Usa i camerieri 40
Imposta i timeout 41

Intercettori di esecuzione 42
Configurazione dell'autenticazione SDK 42
Configura l'accesso alle credenziali temporanee 42
Catena di fornitori di credenziali predefinita 45
Utilizza un provider di credenziali specifico 46
Usa i profili 49
Carica credenziali temporanee da un processo esterno 52
Fornisci credenziali temporanee in codice 57
Leggi le credenziali del ruolo IAM su Amazon EC2 60
Usa Regioni AWS 62
configura esplicitamente un Regione AWS 62
Determina la regione dall'ambiente 62
Verifica la disponibilità del servizio 64
Scegli un endpoint specifico 64
Riduci i tempi di avvio dell'SDK per AWS Lambda 64
Utilizzate un client AWS HTTP basato su CRT 65
Rimuovi le dipendenze inutilizzate del client HTTP 65
Configura i client di servizio per abbreviare le ricerche 66
Inizializza il client SDK all'esterno del gestore di funzioni Lambda 67
Riduci al minimo l'iniezione di dipendenza 68
Usa un targeting Maven Archetype AWS Lambda 68
Lambda SnapStart 68
Modifiche alla versione 2.x che influiscono sul tempo di avvio 69
Risorse aggiuntive 69
Client HTTP 69
Client disponibili 69
Consigli per i clienti 70
Impostazioni predefinite intelligenti 76
Configurare il client HTTP basato su Apache 78
Configurare il client HTTP URLConnection basato 83
Configurazione del client HTTP basato su Netty 89
Configurazione AWS dei client HTTP basati su CRT 96
Configurazione dei proxy HTTP 109
Gestione dell'eccezione 115
Perché le eccezioni non sono controllate? 115
AwsServiceException (e sottoclassi) 115

SdkClientException ... 116
Eccezioni e comportamento di ripetizione dei tentativi ... 117
Tentativi ... 117
Strategie di nuovo tentativo ... 117
Specificare una strategia ... 121
Personalizza una strategia ... 123
Migrazione da `RetryPolicy` a `RetryStrategy` ... 125
Implementare `ContentStreamProvider` ... 125
Uso di `mark()` e `reset()` ... 125
Usa il buffering se `mark()` e `reset()` non sono disponibili ... 126
Crea nuovi stream ... 126
Registrazione ... 127
File di configurazione Log4j 2 ... 127
Aggiungi dipendenza dalla registrazione ... 127
Errori e avvisi specifici dell'SDK ... 128
Registrazione riassuntiva di richieste/risposte ... 129
Registrazione SDK a livello di debug ... 130
Abilita la registrazione dei cavi ... 132
Imposta il TTL JVM per le ricerche dei nomi DNS ... 138
Come impostare il TTL JVM ... 138
Best practice ... 139
Riutilizza i client di servizio ... 139
Chiudere i clienti del servizio ... 139
Chiudi i flussi di input ... 139
Ottimizza le configurazioni HTTP ... 140
Usa OpenSSL per Netty ... 140
Configura i timeout delle API ... 140
Usa le metriche ... 142
Risoluzione dei problemi ... 142
Ripristino della connessione ... 142
Timeout di connessione ... 143
Tempo di lettura scaduto ... 143
Timeout del pool di connessioni ... 144
Errori del percorso di classe ... 148
Errori di firma ... 149
Chiusura del pool di connessioni ... 150

Usa le funzionalità dell'SDK 153
Caratteristiche generali 153
Funzionalità specifiche del servizio 153
Lavora con risultati impaginati 153
Impaginazione sincrona 154
Impaginazione asincrona 157
Sondaggio sugli stati delle risorse 162
Prerequisiti 162
Usare i camerieri 163
Configura i camerieri 164
Esempi di codice 165
Usa la programmazione asincrona 165
Usa un client asincrono APIs 165
Gestisci lo streaming con metodi asincroni 168
Configura opzioni asincrone avanzate 172
Lavora con HTTP/2 173
Pubblica le metriche SDK 174
Quali sono le diverse `MetricPublisher` implementazioni? 174
Applicazioni a lunga durata 175
AWS Lambda funzioni 179
Quando sono disponibili le metriche? 182
Quali informazioni vengono raccolte? 182
Come posso usare queste informazioni? 182
Metriche dei client di assistenza 183
Lavora con Servizi AWS 188
CloudWatch 188
Ottieni metriche da CloudWatch 189
Pubblica dati metrici personalizzati su CloudWatch 191
Lavora con gli CloudWatch allarmi 193
Usa Amazon CloudWatch Events 197
AWS servizi di database 201
Amazon DynamoDB 201
Amazon RDS 202
Amazon Redshift 202
Amazon Aurora senza server v1 202
Amazon DocumentDB 203

DynamoDB 203
Utilizza endpoint basati sull' AWS account 203
Lavora con le tabelle in DynamoDB 204
Lavora con gli elementi in DynamoDB 214
Mappa gli oggetti sugli elementi di DynamoDB 221
Amazon EC2 356
Gestione delle Amazon EC2 istanze 357
Zone Regioni AWS di utilizzo e disponibilità 363
Collabora con i gruppi di sicurezza in Amazon EC2 371
Lavora con i metadati delle EC2 istanze Amazon 375
IAM 382
Gestire le chiavi di IAM accesso 382
Gestisci IAM gli utenti 388
Creare policy IAM 393
Lavora con IAM le politiche 400
Lavora con i certificati IAM del server 407
Kinesis 411
Iscriviti a Amazon Kinesis Data Streams 412
Lambda 423
Invocazione di una funzione Lambda. 423
Elenca le funzioni Lambda 424
Eliminazione di una funzione Lambda 425
Amazon S3 426
Client S3 nell'SDK 426
Caricamento di stream su S3 430
Prefirmato URLs 437
Accesso interregionale 445
Protezione dell'integrità dei dati con checksum 446
Usa un client S3 performante 448
Configurare il supporto per il trasferimento parallelo 452
Trasferisci file e directory 454
Notifiche di eventi di Amazon S3 463
Amazon SNS 473
Creazione di un argomento 473
Elenca i tuoi Amazon SNS argomenti 474
Sottoscrizione di un endpoint a un argomento 475

Pubblicazione di un messaggio in un argomento 476
Annullamento della sottoscrizione di un endpoint da un argomento 477
Eliminazione di un argomento 478
Amazon SQS 479
Utilizza il raggruppamento automatico delle richieste 479
Operazioni di coda 485
Operazioni relative ai messaggi 488
Amazon Transcribe 492
Configura il microfono 492
Crea un editore 492
Crea il client e avvia lo streaming 495
Ulteriori informazioni 491
Esempi di codice 498
ACM 501
Azioni 501
API Gateway 519
Azioni 501
Scenari 523
AWS contributi della comunità 525
Application Auto Scaling 525
Azioni 501
Application Recovery Controller 534
Azioni 501
Aurora 536
Nozioni di base 538
Azioni 501
Scenari 523
Auto Scaling 572
Nozioni di base 538
Azioni 501
Scenari 523
AWS Batch 635
Nozioni di base 538
Azioni 501
Amazon Bedrock 684
Azioni 501

Runtime di Amazon Bedrock 689
Scenari 523
AI21 Laboratori Jurassic-2 707
Amazon Nova 713
Amazon Nova Tela 733
Amazon Titan Image Generator 736
Testo Amazon Titan 739
Amazon Titan Text Embeddings 749
Anthropic Claude 753
Cohere Command 772
Meta Lama 787
IA Mistral 798
Diffusione stabile 809
CloudFront 812
Azioni 501
Scenari 523
CloudWatch 832
Nozioni di base 538
Azioni 501
Scenari 523
CloudWatch Eventi 908
Azioni 501
CloudWatch Registri 914
Azioni 501
Scenari 523
Amazon Cognito Identity 925
Azioni 501
Provider di identità Amazon Cognito 932
Azioni 501
Scenari 523
Amazon Comprehend 959
Azioni 501
Scenari 523
Firehose 972
Azioni 501
Scenari 523

Amazon DocumentDB 982
Esempi serverless 982
DynamoDB 983
Nozioni di base 538
Azioni 501
Scenari 523
Esempi serverless 982
AWS contributi della comunità 525
Amazon EC2 1219
Nozioni di base 538
Azioni 501
Scenari 523
Amazon ECR 1317
Nozioni di base 538
Azioni 501
Amazon ECS 1356
Azioni 501
Elastic Load Balancing - Versione 2 1370
Azioni 501
Scenari 523
MediaStore 1414
Azioni 501
AWS Entity Resolution 1429
Nozioni di base 538
Azioni 501
OpenSearch Servizio 1478
Nozioni di base 538
Azioni 501
EventBridge 1507
Nozioni di base 538
Azioni 501
Scenari 523
EventBridge Scheduler 1542
Azioni 501
Scenari 523
Previsione 1567

Azioni 501
AWS Glue 1580
Nozioni di base 538
Azioni 501
HealthImaging 1612
Azioni 501
Scenari 523
IAM 1643
Nozioni di base 538
Azioni 501
Scenari 523
AWS IoT 1727
Nozioni di base 538
Azioni 501
AWS IoT data 1767
Azioni 501
AWS IoT FleetWise 1770
Nozioni di base 538
Azioni 501
AWS IoT SiteWise 1823
Nozioni di base 538
Azioni 501
Amazon Keyspaces 1869
Nozioni di base 538
Azioni 501
Kinesis 1895
Azioni 501
Esempi serverless 982
AWS KMS 1908
Nozioni di base 538
Azioni 501
Lambda 1965
Nozioni di base 538
Azioni 501
Scenari 523
Esempi serverless 982

AWS contributi della comunità 525
Amazon Lex 2002
Scenari 523
Amazon Location 2003
Nozioni di base 538
Azioni 501
Location Service Places 2050
Azioni 501
Marketplace AWS API del catalogo 2055
Prodotti AMI 2055
Offerte per i partner di canale 2081
Prodotti container 2097
Entità 2103
Offers (Offerte) 2109
Prodotti 2168
Autorizzazione alla rivendita 2174
Prodotti SaaS 2215
Utilità 2241
Marketplace AWS API di accordo 2245
Accordi 2245
MediaConvert 2299
Azioni 501
Migration Hub 2321
Azioni 501
MSK Amazon 2334
Esempi serverless 982
Neptune 2336
Nozioni di base 538
Azioni 501
Scenari 523
Partner Central 2378
Azioni 501
Scenari 523
Amazon Personalize 2414
Azioni 501
Eventi di Amazon Personalize 2444

Azioni 501
Runtime di Amazon Personalize 2447
Azioni 501
Amazon Pinpoint 2452
Azioni 501
API SMS e Voce di Amazon Pinpoint 2496
Azioni 501
Amazon Polly 2500
Azioni 501
Scenari 523
Amazon RDS 2507
Nozioni di base 538
Azioni 501
Scenari 523
Esempi serverless 982
Servizi di dati di Amazon RDS 2549
Scenari 523
Amazon Redshift 2551
Nozioni di base 538
Azioni 501
Scenari 523
Amazon Rekognition 2589
Azioni 501
Scenari 523
Registrazione del dominio Route 53 2661
Nozioni di base 538
Azioni 501
Amazon S3 2683
Nozioni di base 538
Azioni 501
Scenari 523
Esempi serverless 982
Controllo Amazon S3 2868
Nozioni di base 538
Azioni 501
Bucket di directory S3 2911

Nozioni di base 538
Azioni 501
Scenari 523
S3 Glacier 2993
Azioni 501
SageMaker AI 3009
Azioni 501
Scenari 523
Secrets Manager 3038
Azioni 501
Amazon SES 3041
Azioni 501
Scenari 523
API Amazon SES v2 3057
Azioni 501
Scenari 523
Amazon SNS 3077
Azioni 501
Scenari 523
Esempi serverless 982
Amazon SQS 3147
Azioni 501
Scenari 523
Esempi serverless 982
Step Functions 3221
Nozioni di base 538
Azioni 501
Scenari 523
AWS STS 3246
Azioni 501
Supporto 3249
Nozioni di base 538
Azioni 501
Systems Manager 3272
Nozioni di base 538
Azioni 501

Amazon Textract ... 3315
Azioni ... 501
Scenari ... 523
Amazon Transcribe ... 3327
Azioni ... 501
Scenari ... 523
Streaming Amazon Transcribe ... 3338
Azioni ... 501
Scenari ... 523
Amazon Translate ... 3357
Scenari ... 523
Sicurezza ... 3360
Protezione dei dati ... 3360
Transport Layer Security (TLS) ... 3361
Controlla le versioni TLS ... 3362
Applica le versioni TLS ... 3363
Migrazione a TLS 1.2 ... 3363
Identity and Access Management ... 3363
Destinatari ... 3364
Autenticazione con identità ... 3364
Gestione dell'accesso con policy ... 3368
Come Servizi AWS lavorare con IAM ... 3370
Risoluzione dei problemi di AWS identità e accesso ... 3371
Convalida della conformità ... 3373
Resilienza ... 3374
Sicurezza dell'infrastruttura ... 3374
Migrare alla versione 2 ... 3376
Novità della versione 2 ... 3376
Come migrare ... 3377
Strumento di migrazione (versione di anteprima) ... 3377
Step-by-step istruzioni ... 3383
Cosa c'è di diverso tra 1.x e 2.x ... 3402
Modifica del nome del pacchetto ... 3402
Aggiungere la versione 2.x al progetto ... 3403
Immutabile POJOs ... 3403
Metodi Setter e Getter ... 3404

nomi delle classi del modello 3404
Biblioteche e servizi di pubblica utilità 3405
Modifiche al client 3407
Modifiche al provider delle credenziali 3452
Modifiche alla regione 3461
Modifiche alle operazioni, alle richieste e alle risposte 3463
Modifiche alle eccezioni 3470
Modifiche specifiche del servizio 3471
Lavorare con Amazon S3 3477
Modifiche al file di profilo 3510
Configurazione esterna 3511
Waiter 3514
EC2 utilità per i metadati 3518
CloudFront preassegnazione 3526
API IAM Policy Builder 3530
Mappatura/documento DynamoDB APIs 3536
Suddivisione automatica delle richieste SQS 3566
Usa l'SDK per Java 1.x e 2.x side-by-side 3575
Chiave OpenPGP 3577
Chiave attuale 3577
Chiavi storiche 3581
Cronologia dei documenti 3584
..... mmmdxcvi

Che cos'è il AWS SDK for Java 2.x

AWS SDK per Java Fornisce un'API Java per Servizi AWS. Utilizzando l'SDK, è possibile creare applicazioni Java che funzionano con Amazon S3, Amazon EC2 DynamoDB, e altro ancora.

AWS SDK for Java 2.x Si tratta di un'importante riscrittura 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 per 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. Nozioni di base

Per configurare il tuo ambiente di sviluppo, consulta. the section called “Configurazione”

Se attualmente utilizzi la versione 1.x di SDK per Java, consulta Migrare alla versione 2 per indicazioni specifiche.

Per informazioni su come effettuare richieste a Amazon S3, Amazon EC2 e altro DynamoDB Servizi AWS, consulta Use the SDK per Java and Work with. Servizi AWS

Sviluppa applicazioni mobili

Se sei uno sviluppatore di app per dispositivi mobili, Amazon Web Services fornisce il AWS Amplifyframework.

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 AWS SDKs and Tools Reference Guide:

- AWS SDKs e politica di manutenzione degli strumenti
- AWS SDKs e Tools Version Support Matrix

Risorse aggiuntive

Oltre a questa guida, le seguenti sono preziose risorse online per AWS SDK per Java gli sviluppatori:

- AWS SDK per Java Riferimento alle API 2.x
- Blog per gli sviluppatori Java
- Argomento sullo sviluppo di Java in AWS re:Post
- Sorgente SDK attiva GitHub
- AWS Libreria di esempi di codice SDK
- @awsforjava (Twitter)

Contribuisci all'SDK

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

- Invia i problemi relativi all'SDK su GitHub
- Partecipa a una chat informale sull'SDK sul canale gitter 2.x AWS SDK per Java

Guida introduttiva alla versione AWS SDK per Java 2.x

Le sezioni di questo argomento illustrano i passaggi essenziali per iniziare a creare applicazioni Java che si connettono a. Servizio AWS Le sezioni riguardano la configurazione dell'ambiente di sviluppo con Java e la creazione di strumenti come Maven o Gradle, la configurazione dell'autenticazione sicura e la creazione della prima applicazione funzionante tramite un tutorial pratico. AWS Questo argomento, adatto ai principianti, funge da punto di partenza per AWS lo sviluppo con Java e fornisce le basi necessarie prima di esplorare funzionalità più avanzate.

Indice

- Configurazione del AWS SDK per Java 2.x
  - Panoramica della configurazione
  - Installa Java e uno strumento di compilazione per lavorare con AWS SDK for Java 2.x
  - Configura un progetto Apache Maven che utilizza AWS SDK for Java 2.x
    - Prerequisiti
    - Creare un progetto Maven
    - Configurare il compilatore Java per Maven
    - Dichiarare l'SDK come dipendenza
    - Impostare le dipendenze per i moduli SDK
      - Creare l'intero SDK nel progetto
    - Creare il progetto
  - Configura un progetto Gradle che utilizza il AWS SDK for Java 2.x
  - Configura un progetto GraalVM Native Image che utilizza AWS SDK for Java 2.x
    - Prerequisiti
    - Crea un progetto usando l'archetipo
    - Crea un'immagine nativa
- Autenticazione con l'utilizzo di AWSAWS SDK for Java 2.x
  - Configurazione per l'autenticazione
    - 1. Configurazione per l'accesso Single Sign-On per l'SDK
    - 2. Effettuate l'accesso utilizzando il AWS CLI
  - Opzioni di autenticazione aggiuntive
- Creazione di una semplice applicazione utilizzando AWS SDK for Java 2.x

- Fase 1: Configurazione per questo tutorial
- Fase 2: Creare il progetto
- Passaggio 3: scrivere il codice
- Fase 4: Compilare ed eseguire l'applicazione
  - Riuscito
  - Rimozione
- Passaggi successivi

Configurazione del AWS SDK per 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 per 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' Servizi 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

Se utilizzi IntelliJ IDEA, puoi anche aggiungere Kit di strumenti AWS per IntelliJ IDEA da integrare Servizi AWS direttamente nell'IDE per aiutarti a semplificare lo sviluppo.

- 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

Installa Java e uno strumento di compilazione per lavorare con AWS SDK for Java 2.x

Sono necessari i seguenti requisiti dell'ambiente di sviluppo Java per lavorare con SDK for Java 2.x:

- Java 8 o versione successiva. AWS SDK per 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.
- Strumenti di compilazione compatibili con Maven come Apache Maven, Apache Ant with Ivy, Gradle o IntelliJ.
  - Per informazioni su come installare e utilizzare Maven, consulta https://maven.apache.org/.
  - Per informazioni su come installare e utilizzare Apache Ivy, vedi https://ant.apache.org/ivy/.
  - Per informazioni su come installare e utilizzare Gradle, consulta https://gradle.org/.
  - Per informazioni su come installare e utilizzare IntelliJ IDEA, vedere. https://www.jetbrains.com/idea/

Configura un progetto Apache Maven che utilizza AWS SDK for Java 2.x

Puoi usare Apache Maven per configurare e creare AWS SDK for Java 2.x progetti o per creare l'SDK stesso.

Prerequisiti

Per utilizzare l'SDK for Java 2.x con Maven, è necessario quanto segue:

- Java 8.0 o versione successiva. È possibile scaricare il software Java SE Development Kit più recente da http://www.oracle. com/technetwork/java/javase/downloads/. L'SDK for Java 2.x funziona anche con OpenJDK Amazon Corretto e una distribuzione dell'Open Java Development Kit (OpenJDK). Scarica l'ultima versione 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, sostituiscila *2.X.X* con la 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 utilizzarlo AWS SDK per Java nel progetto, è necessario dichiararlo come dipendenza nel file del progetto. `pom.xml`

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>
```

> **Note**
>
> Sostituisci il *2.X.X*`pom.xml` file con la versione più recente di. AWS SDK for Java 2.x

Impostare le dipendenze per i moduli SDK

Ora che hai configurato l'SDK, puoi aggiungere dipendenze per uno o più AWS SDK per Java moduli da utilizzare nel tuo progetto.

Sebbene sia possibile specificare il numero di versione per ogni componente, non è necessario perché la versione 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 di`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`o`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 per 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 sotto/.) 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 inserire il tutto AWS SDK per Java nel tuo progetto, dichiaralo nel tuo `pom.xml` file, 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 che utilizza il AWS SDK for Java 2.x

Puoi usare Gradle per configurare e creare progetti. AWS SDK for Java 2.x

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'`init`attività, 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 strutturato.

```
/*
 * 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:33.3.0-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:2.27.21"))
       // With the bom declared, you specify individual SDK dependencies without a
    version.
    ...
   }
   ...
   ```

   > **Note**
   >
   > In questo file di build di esempio, sostituisci 2.27.21 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:2.27.21"))
       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 sia in Kotlin che in Groovy. DSLs 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:ssooidc")
    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:2.27.21')
    implementation 'software.amazon.awssdk:s3'
    implementation 'software.amazon.awssdk:sso'
    implementation 'software.amazon.awssdk:ssooidc'
    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.

Configura un progetto GraalVM Native Image che utilizza AWS SDK for Java 2.x

Con le versioni 2.16.1 e successive, AWS SDK for Java 2.x 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 di immagini nativo integrato.

Prerequisiti

- Completa i passaggi descritti in Configurazione del 2.x. AWS SDK per Java
- Installa GraalVM Native Image.

Crea un progetto usando l'archetipo

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

> **Note**
>
> `com.example.mynativeimageapp`Sostituiscilo con lo spazio dei nomi completo del pacchetto dell'applicazione. Sostituiscilo anche `mynativeimageapp` con il nome del tuo progetto. Questo sarà il nome della directory per il progetto.

```
mvn archetype:generate \
    -DarchetypeGroupId=software.amazon.awssdk \
    -DarchetypeArtifactId=archetype-app-quickstart \
    -DarchetypeVersion=2.27.21\
    -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 AWS SDK per Java Amazon S3, e il `ApacheHttpClient` client HTTP. Include anche una dipendenza per il plug-in GraalVM Native Image Maven, in modo da poter creare immagini native utilizzando Maven.

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

Per utilizzare un client HTTP `-DhttpClient` `netty-nio-client` asincrono, imposta il parametro su. Per utilizzarlo invece `UrlConnectionHttpClient` come client HTTP sincronoapache-client, imposta il parametro su. `-DhttpClient` `url-connection-client`

Crea un'immagine nativa

Dopo aver creato il progetto, esegui il seguente comando dalla directory del progetto, ad esempiomynativeimageapp:

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

In questo modo viene creata un'applicazione di immagini nativa nella `target` directory, ad esempio`target/mynativeimageapp`.

Autenticazione con l'utilizzo di AWSAWS SDK for Java 2.x

Quando si utilizza la AWS SDK for Java 2.x, una cosa importante da sapere sull'autenticazione è che l'SDK gestisce automaticamente il complesso processo di firma delle richieste utilizzando le credenziali dell'ambiente o dei ruoli IAM senza richiedere l'implementazione di algoritmi crittografici.

L'SDK gestisce l'individuazione delle credenziali, la creazione delle firme e l'aggiornamento delle credenziali completamente in background, consentendoti di concentrarti sulla logica dell'applicazione.

Configurazione per l'autenticazione

L'argomento Autenticazione e accesso nella Guida di riferimento agli strumenti AWS SDKs e agli strumenti descrive i diversi approcci 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 contenute nella AWS SDKs Tools Reference Guide, il sistema dovrebbe essere configurato in modo da consentire all'SDK di firmare le richieste:

1. 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 valore `sso_role_name`, che è un ruolo IAM connesso a un set di autorizzazioni di IAM Identity Center, dovrebbe consentire l'accesso ai 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 sezione `sso-session` contiene le impostazioni per avviare una sessione del portale di accesso AWS .

```
[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 AWS SDKs and Tools Reference Guide.

Se l'ambiente di sviluppo non è configurato per l'accesso programmatico come mostrato in precedenza, segui il passaggio 2 della SDKs Guida di riferimento.

2. Effettuate l'accesso 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

```
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 seguente comando AWS CLI .

```
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 eseguiaws 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. botocoreè la base per AWS CLI .
> Seleziona Consenti per autorizzare l'accesso alle tue informazioni per AWS CLI e SDK for Java.

Opzioni di autenticazione aggiuntive

Per ulteriori opzioni sull'autenticazione per l'SDK, come l'uso di profili e variabili di ambiente, consultate il capitolo sulla configurazione nella Guida di riferimento agli strumenti AWS SDKs e agli strumenti.

Creazione di una semplice applicazione utilizzando AWS SDK for Java 2.x

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=2.27.21
```

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`

Prompt	Valore da inserire
`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 *Y*o 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>2.27.21</aws.java.sdk.version> <-------- SDK version
 picked up from archetype version.
        <slf4j.version>1.7.28</slf4j.version>
        <junit5.version>5.8.1</junit5.version>
    </properties>

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

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

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

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

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

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

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

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

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

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

</project>
```

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`Handler.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 DynamoDBAmazon EC2, e vari servizi di database
- Usa l'SDK
- Sicurezza per AWS SDK per Java

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 e altri. EC2

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 l'opzione `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();
```

Chiudere il client di 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'`try`istruzione -with-resources. I client di servizio implementano l'`Autoclosable`interfaccia, quindi il JDK chiama automaticamente il metodo alla fine dell'`close()`istruzione.

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();
    }
}
```

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 un'API fluida che puoi utilizzare per creare una richiesta. Con l'API fluent 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));
```

Utilizzate le richieste per sovrascrivere la configurazione del client

Sebbene un client di servizio sia immutabile, è possibile sovrascrivere molte delle sue impostazioni a livello di richiesta. Quando si crea una richiesta, è possibile fornire un'AwsRequestOverrideConfigurationistanza per fornire le impostazioni sostituite. Alcuni dei metodi che puoi usare per sovrascrivere le impostazioni del client sono:

- `apiCallAttemptTimeout`
- `apiCallTimeout`
- `credentialProvider`
- `compressionConfiguration`
- `putHeader`

Per un esempio di sovrascrittura di un'impostazione del client con una richiesta, supponiamo di avere il seguente client S3 che utilizza le impostazioni predefinite.

```
S3Client s3Client = S3Client.create();
```

Vuoi scaricare un file di grandi dimensioni e assicurarti che la richiesta non scada prima del termine del download. A tale scopo, aumenta i valori di timeout per una sola `GetObject` richiesta, come mostrato nel codice seguente.

Standard API

```
AwsRequestOverrideConfiguration overrideConfiguration =
 AwsRequestOverrideConfiguration.builder()
    .apiCallTimeout(Duration.ofSeconds(100L))
    .apiCallAttemptTimeout(Duration.ofSeconds(25L))
    .build();

GetObjectRequest request = GetObjectRequest.builder()
    .bucket("DOC-EXAMPLE-BUCKET")
    .key("DOC-EXAMPLE-KEY")
    .overrideConfiguration(overrideConfiguration)
    .build();

s3Client.getObject(request, myPath);
```

Fluent API

```
s3Client.getObject(b -> b
        .bucket("DOC-EXAMPLE-BUCKET")
        .key("DOC-EXAMPLE-KEY")
        .overrideConfiguration(c -> c
            .apiCallTimeout(Duration.ofSeconds(100L))
            .apiCallAttemptTimeout(Duration.ofSeconds(25L))),
    myPath);
```

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());
}
```

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`o 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 (SdkClientExeption 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, vedi Uso dei camerieri.

Imposta i timeout

Puoi configurare i timeout per ciascuno dei tuoi client di servizio utilizzando i `apiCallTimeout`e i `apiCallAttemptTimeout`setter di. `ClientOverrideConfiguration.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`e 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 default 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'`ExecutionInterceptor`interfaccia nell'API Reference. AWS SDK per Java

Configurazione dell'autenticazione 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 provider di credenziali specifico
- Usa i profili
- Carica credenziali temporanee da un processo esterno
- Fornisci credenziali temporanee in 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 chiave di accesso e chiave di accesso segreta) e 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 sull'autenticazione della and Tools Reference Guide. AWS SDKs

> **Note**
>
> Oltre alla configurazione impostata nel `config` file che funziona per tutti i progetti, ogni singolo progetto Java richiede che il `pom.xml` file Maven contenga le seguenti dipendenze:
>
> ```
> <dependency>
>     <groupId>software.amazon.awssdk</groupId>
>     <artifactId>sso</artifactId>
> </dependency>
> <dependency>
>     <groupId>software.amazon.awssdk</groupId>
>     <artifactId>ssooidc</artifactId>
> </dependency>
> ```
>
> Le `ssooidc` dipendenze `sso` e forniscono il codice che consente all'SDK for Java 2.x di accedere alle credenziali temporanee.

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>
aws_session_token=<value from AWS access portal>
```

3. Salvare il file. Il file `~/.aws/credentials` dovrebbe ora esistere nel sistema di sviluppo locale. Questo file contiene il profilo [predefinito] utilizzato dall'SDK for Java se non viene specificato un profilo denominato specifico.
4. Accedi al portale di AWS accesso
5. Segui queste istruzioni sotto l'intestazione Aggiornamento manuale delle credenziali per copiare le credenziali del ruolo IAM dal AWS portale di accesso.
   a. Per la fase 2 delle istruzioni collegate, scegli `Access keys` il nome del ruolo IAM che concede l'accesso per le tue esigenze di sviluppo. Questo ruolo in genere ha un nome simile PowerUserAccessa Developer.
   b. Nella finestra di dialogo modale, selezionate il sistema operativo in uso e copiate il contenuto da Aggiungi un profilo al file delle AWS credenziali.
6. Incolla le credenziali copiate nel `credentials` file locale e rimuovi il nome del profilo generato. Il file dovrebbe essere simile al seguente.

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

7. Salvare il file `credentials`.

Quando l'SDK for Java crea un client di servizio, accederà a queste credenziali temporanee e le utilizzerà per ogni richiesta. Le impostazioni per il ruolo IAM scelto nella fase 5a determinano la durata di validità delle credenziali temporanee. La durata massima è di dodici ore.

Dopo la scadenza delle credenziali temporanee, ripeti i passaggi da 4 a 7.

Catena di fornitori di credenziali predefinita

La catena di fornitori di credenziali predefinita è implementata dalla DefaultCredentialsProviderclasse. Controlla in sequenza ogni posizione in cui è possibile impostare la configurazione predefinita per la fornitura di credenziali temporanee, quindi seleziona la prima che imposti.

Per utilizzare la catena di fornitori di credenziali predefinita per fornire credenziali temporanee, create un service client builder ma non specificate un provider di credenziali. Il seguente frammento di 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 SystemPropertyCredentialsProviderclasse per caricare credenziali temporanee dalle proprietà di sistema `aws.accessKeyIdaws.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 EnvironmentVariableCredentialsProviderclasse per caricare credenziali temporanee dalle variabili di ambiente `AWS_ACCESS_KEY_IDAWS_SECRET_ACCESS_KEY`, e. `AWS_SESSION_TOKEN`
3. Token di identità Web da AWS Security Token Service

- L'SDK utilizza la WebIdentityTokenFileCredentialsProviderclasse 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 ProfileCredentialsProviderper caricare le impostazioni Single Sign-on o le credenziali temporanee di IAM Identity Center dal `[default]` profilo nei file condivisi e. `credentials` `config`

     La AWS SDKs and Tools Reference Guide 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` e sono condivisi da vari strumenti. AWS SDKs Per ulteriori informazioni, vedere The. aws/credentials and .aws/configfile nella AWS SDKs and Tools Reference Guide.

5. Amazon ECS credenziali del contenitore

   - L'SDK utilizza la ContainerCredentialsProviderclasse per caricare credenziali temporanee dalle seguenti variabili di ambiente:

     `AWS_CONTAINER_CREDENTIALS_RELATIVE_URI` o `AWS_CONTAINER_CREDENTIALS_FULL_URI`

     `AWS_CONTAINER_AUTHORIZATION_TOKEN_FILE` o `AWS_CONTAINER_AUTHORIZATION_TOKEN`

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

   - L'SDK utilizza la InstanceProfileCredentialsProviderclasse per caricare credenziali temporanee dal servizio di metadati. Amazon EC2

Utilizza un provider di credenziali specifico

L'SDK utilizza i fornitori di credenziali per recuperare, gestire e fornire le credenziali di autenticazione (come chiavi di accesso e token di sessione) necessarie per l'accesso. Servizi AWS

I fornitori di credenziali semplificano il recupero delle credenziali da varie fonti, implementano le migliori pratiche di sicurezza e supportano strategie di autenticazione flessibili in tutti gli ambienti. AWS

Specificare un fornitore di credenziali

Per aggirare la catena di provider di credenziali predefinita, specifica quale provider di credenziali deve utilizzare un client di servizio. 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 EnvironmentVariableCredentialsProvideroggetto al `credentialsProvider` metodo sul service client builder, come mostrato 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 Implementing Classes nel riferimento API per. AwsCredentialsProvider

> **Note**
>
> Puoi anche utilizzare il tuo fornitore di credenziali o le tue catene di provider implementando l'`AwsCredentialsProvider`interfaccia.

Configurare un provider di credenziali

Come esempio di configurazione di un'implementazione di un provider di credenziali, potresti voler fare in modo che l'SDK utilizzi un thread in background per prerecuperare (recuperare in anticipo) le credenziali prima che scadano. In questo modo puoi evitare la chiamata di blocco che recupera nuove credenziali.

Di seguito viene mostrato un esempio che crea un file `StsAssumeRoleCredentialsProvider` che utilizza un thread in background per prerecuperare le credenziali impostando la `asyncCredentialUpdateEnabled` proprietà su on the builder: `true`

```
S3Client s3Client = S3Client.builder()
    .credentialsProvider(StsAssumeRoleCredentialsProvider.builder()
        .asyncCredentialUpdateEnabled(true)
        .stsClient(StsClient.create())
        .refreshRequest(r -> r
            .roleArn("arn:aws:iam::111122223333:role/S3-listbuckets-only-role")
            .roleSessionName("test-temp-session")
            .durationSeconds(900))
        .build())
    .build();
```

Quando si richiama un'operazione on `s3Client` per la prima volta, un `AssumeRoleRequest` viene inviato a (STS). AWS Security Token Service STS restituisce credenziali temporanee valide per 15 minuti (900 secondi). L'`s3Client`istanza utilizza le credenziali memorizzate nella cache fino al momento di aggiornarle prima che trascorrano i 15 minuti. Per impostazione predefinita, l'SDK tenta di recuperare nuove credenziali per una nuova sessione tra 5 minuti e 1 minuto prima della scadenza della sessione corrente. La finestra di pre-fetch è configurabile utilizzando le proprietà and. `prefetchTime` `staleTime`

È possibile configurare in modo analogo i seguenti provider di credenziali basati sulla sessione:

- `StsWebIdentityTokenFileCredentialsProvider`
- `StsGetSessionTokenCredentialsProvider`
- `StsGetFederationTokenCredentialsProvider`
- `StsAssumeRoleWithWebIdentityCredentialsProvider`
- `StsAssumeRoleWithSamlCredentialsProvider`
- `StsAssumeRoleCredentialsProvider`
- `DefaultCredentialsProvider`(quando delega al provider di credenziali che utilizza le sessioni)
- `ProcessCredentialsProvider`
- `WebIdentityTokenFileCredentialsProvider`
- `ContainerCredentialsProvider`
- `InstanceProfileCredentialsProvider`

Usa i profili

Utilizzando il `credentials` file condivisoconfig, 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 ProfileCredentialsProviderclasse 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:, , e `ProfileCredentialsProvider` `DefaultCredentialsProvider` `InstanceProfileCredentialsProvider` `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_sessionsso_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 numerose 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` (o`config`).

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. SDKs e AWS CLI può 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, consultate la sezione relativa alle credenziali di processo nella and Tools Reference Guide. AWS SDKs

Utilizzate un **`ProcessCredentialsProvider`**

In alternativa all'utilizzo delle impostazioni nel `config` file, puoi utilizzare gli SDK `ProcessCredentialsProvider` per caricare le 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();
```

Usa IAM Roles Anywhere per l'autenticazione

IAM Roles Anywhere Servizio AWS ti consente di ottenere AWS credenziali temporanee per carichi di lavoro eseguiti al di fuori di. AWS Consente l'accesso sicuro alle AWS risorse da ambienti locali o altri ambienti cloud.

Prima di poter autenticare le richieste con IAM Roles Anywhere, devi prima raccogliere le informazioni richieste e scaricare lo strumento di supporto alle credenziali. Seguendo le istruzioni introduttive nella Guida per l'utente di IAM Roles Anywhere, puoi creare gli elementi necessari.

L'SDK for Java non dispone di un provider di credenziali dedicato per recuperare credenziali temporanee da IAM Roles Anywhere, ma puoi utilizzare lo strumento di supporto delle credenziali insieme a una delle opzioni per recuperare le credenziali da un processo esterno.

Usa l'impostazione in un profilo **`credential_process`**

Il seguente frammento del file di AWS configurazione condiviso mostra un profilo denominato `roles_anywhere` che utilizza l'impostazione: `credential_process`

```
[profile roles_anywhere]
```

```
credential_process = ./aws_signing_helper credential-process \
  --certificate /path/to/certificate \
  --private-key /path/to/private-key \
  --trust-anchor-arn arn:aws:rolesanywhere:region:account:trust-anchor/TA_ID \
  --profile-arn arn:aws:rolesanywhere:region:account:profile/PROFILE_ID \
  --role-arn arn:aws:iam::account:role/role-name-with-path
```

È necessario sostituire il testo mostrato in rosso con i valori dopo aver assemblato tutti gli artefatti. Il primo elemento dell'impostazione,`aws_signing_helper`, è l'eseguibile dello strumento di supporto alle credenziali ed `credential-process` è il comando.

Quando configuri un client di servizio per l'utilizzo del `roles_anywhere` profilo, come illustrato nel codice seguente, l'SDK memorizza nella cache le credenziali temporanee e le aggiorna prima della scadenza:

```
S3Client s3Client = S3Client.builder()
    .credentialsProvider(ProfileCredentialsProvider.builder()
        .profileName("roles_anywhere").build())
    .build();
```

Configura un **`ProcessCredentialsProvider`**

Come illustrato di seguito, puoi utilizzare un approccio basato solo sul codice `ProcessCredentialsProvider` anziché utilizzare le impostazioni del profilo:

```
ProcessCredentialsProvider processCredentialsProvider =
 ProcessCredentialsProvider.builder()
    .command("""
            ./aws_signing_helper credential-process \
            --certificate /path/to/certificate \
            --private-key /path/to/private-key \
            --trust-anchor-arn arn:aws:rolesanywhere:region:account:trust-anchor/TA_ID
 \
            --profile-arn arn:aws:rolesanywhere:region:account:profile/PROFILE_ID \
            --role-arn arn:aws:iam::account:role/role-name-with-path
        """).build();

S3Client s3Client = S3Client.builder()
    .credentialsProvider(processCredentialsProvider)
    .build();
```

Sostituisci il testo mostrato in rosso con i tuoi valori dopo aver assemblato tutti gli artefatti.

Fornisci credenziali temporanee in 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 (). 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 StaticCredentialsProvideroggetto 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 assunto

Politica sulle autorizzazioni relative ai ruoli assunti

La seguente politica di autorizzazioni è allegata 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' EC2 istanza e che AWS CLI effettuano richieste AWS API. Ciò è preferibile alla memorizzazione delle chiavi di accesso all'interno dell' EC2 istanza. Per assegnare un AWS ruolo a un' EC2 istanza e renderlo disponibile per tutte le sue applicazioni, create un profilo di istanza collegato all'istanza. Un profilo di istanza contiene il ruolo e consente ai programmi in esecuzione sull' EC2 istanza di ottenere credenziali temporanee. Per ulteriori informazioni, consulta Utilizzare un ruolo IAM per concedere le autorizzazioni alle applicazioni in esecuzione su EC2 istanze Amazon nella IAM User Guide.

Questo argomento fornisce informazioni su come configurare l'applicazione Java per l'esecuzione su un' EC2istanza e abilitare l'SDK for Java all' IAM acquisizione delle 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 questo passaggio, l'SDK utilizza un `InstanceProfileCredentialsProvider` per leggere il ruolo IAM definito nel profilo dell' EC2 istanza. 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 `InstanceProfileCredentialsProvider` on EC2, è possibile 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, EC2 le istanze eseguono IMDS (Instance Metadata Service) che consente agli SDK di accedere `InstanceProfileCredentialsProvider` a informazioni come il ruolo IAM che è stato configurato. EC2 per impostazione predefinita, le istanze eseguono due versioni di IMDS:

- Instance Metadata Service Version 1 (IMDSv1): un metodo di richiesta/risposta
- Instance Metadata Service Version 2 (IMDSv2): un metodo orientato alla sessione

IMDSv2 è un approccio più sicuro di. IMDSv1

Per impostazione predefinita, Java SDK tenta innanzitutto di IMDSv2 ottenere il ruolo IAM, ma se fallisce, ci prova IMDSv1. Tuttavia, poiché IMDSv1 è meno sicuro, AWS consiglia di utilizzare IMDSv2 solo l'SDK e di disabilitarne i tentativi. IMDSv1

Per utilizzare un approccio più sicuro, disabilita l'utilizzo dell'SDK 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 iniziale fallisce. IMDSv2

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.

Quando si sviluppano 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, usa 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>2.27.21</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 delle credenziali dell'SDK.

> **Note**
>
> L'utilizzo di questo provider di credenziali consente di utilizzare il codice nelle Lambda funzioni, ma potrebbe non funzionare su altri sistemi. Amazon EC2
> Se a un certo punto intendi utilizzare Lambda SnapStart per Java, dovresti fare affidamento sulla catena di provider di credenziali predefinita per cercare le credenziali. Se specificate la`EnvironmentVariableCredentialsProvider`, la ricerca iniziale delle credenziali funziona, ma quando SnapStart è attivata, il runtime Java imposta le variabili di ambiente delle credenziali del contenitore. Al momento dell'attivazione, le variabili di ambiente utilizzate dalle variabili di ambiente `EnvironmentVariableCredentialsProvider` —access key— non sono disponibili per Java SDK.

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, l'istanza inizializzata `S3Client` viene riutilizzata.

```
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 SdkHttpClientl'interfaccia. Un client di servizio sincrono, come il `S3Client` o il`DynamoDbClient`, richiede l'uso di un client HTTP sincrono. AWS SDK per Java Offre tre client HTTP sincroni.

ApacheHttpClient (impostazione predefinita)

ApacheHttpClientè 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

AwsCrtHttpClientfornisce 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. UrlConnectionHttpClient Per informazioni sulla configurazione di`UrlConnectionHttpClient`, vedere. Configurare il client HTTP URLConnection basato

Client asincroni

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

NettyNioAsyncHttpClient (impostazione predefinita)

NettyNioAsyncHttpClientè il client HTTP predefinito utilizzato dai client asincroni. Per informazioni sulla configurazione di, vedere. `NettyNioAsyncHttpClient` the section called “Configurazione del client HTTP basato su Netty”

AwsCrtAsyncHttpClient

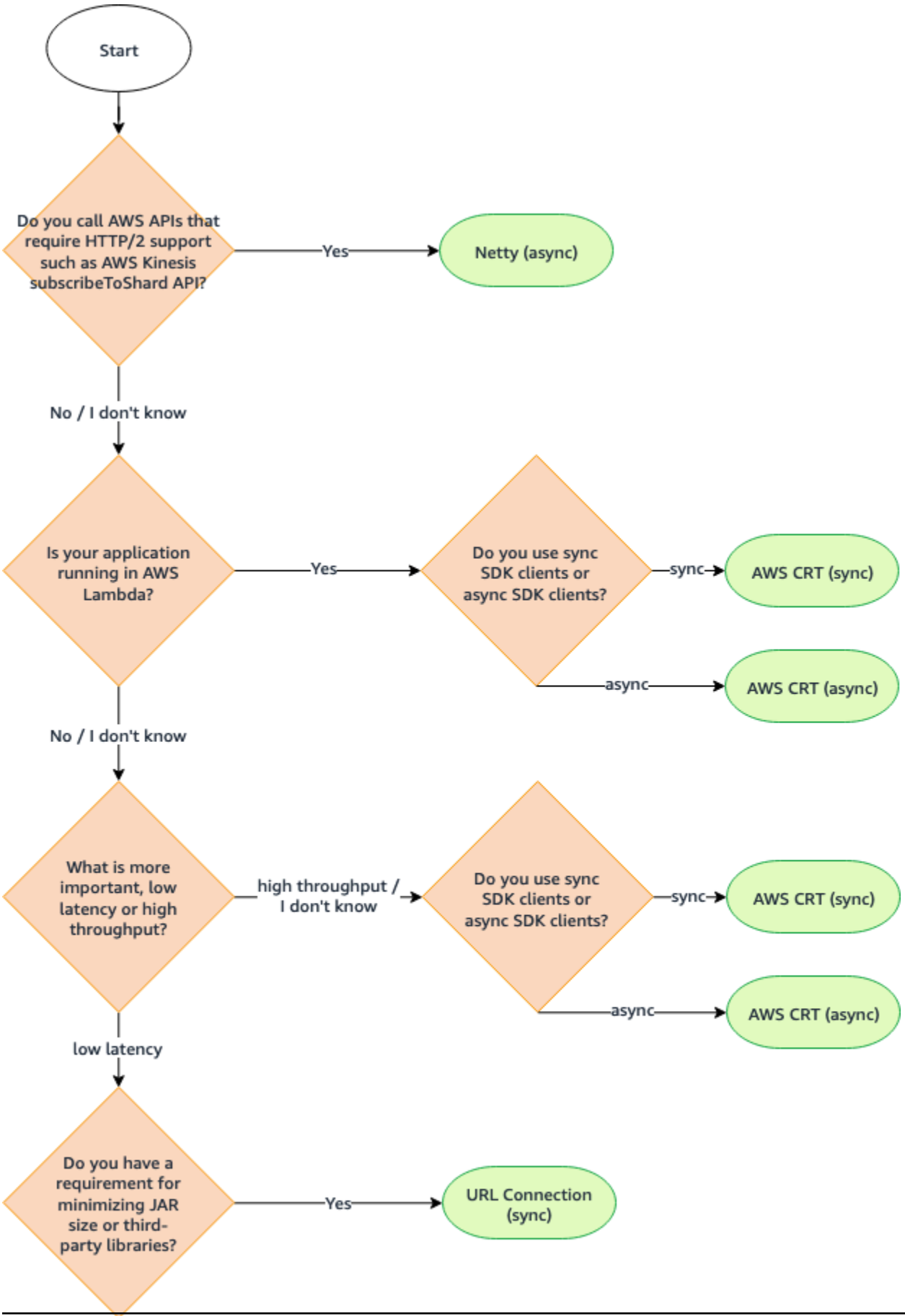
AwsCrtAsyncHttpClientSi basa sul client HTTP AWS Common Runtime (CRT). Per informazioni sulla configurazione di`AwsCrtAsyncHttpClient`, vedere. the section called “Configurazione AWS dei client HTTP basati su CRT”

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.


Start
Do you call AWS APIs that require HTTP/2 support such as AWS Kinesis subscribeToShard API?
Yes
Netty (async)
No / I don't know
Is your application running in AWS Lambda?
Yes
Do you use sync SDK clients or async SDK clients?
sync
AWS CRT (sync)
async
AWS CRT (async)
No / I don't know
What is more important, low latency or high throughput?
high throughput / I don't know
Do you use sync SDK clients or async SDK clients?
sync
AWS CRT (sync)
async
AWS CRT (async)
low latency
Do you have a requirement for minimizing JAR size or third-party libraries?
Yes
URL Connection (sync)
No / I don't know

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
URLConnectionclient HTTP basato su	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	• Usalo se l'applicazione è in esecuzione in AWS Lambda • Usalo se preferisci un throughput elevato rispetto a una bassa latenza • Usalo se preferisci sincronizzare i client SDK	Le seguenti proprietà del sistema Java non sono supportate: • javax.net.ssl.keystore • javax.net.ssl.keyStorePassword • javax.net.ssl.truststore

Client HTTP	Sincronizzazione o asincrona	Quando usare	Limitazione/ inconveniente
			• javax.net. ssl. trustStor ePassword
Client HTTP basato su Netty (client HTTP asincrono predefinito)	Asincrono	• Usalo se l'applicazione richiama richiami APIs che richiedono il supporto HTTP/2 come l'API Kinesis SubscribeToShard	Tempo di avvio più lento rispetto ad altri client HTTP

Client HTTP	Sincronizzazione o asincrona	Quando usare	Limitazione/ inconveniente
AWS Client HTTP asincrono basato su CRT 1	Asincrono	• Usalo se l'applicazione è in esecuzione in AWS Lambda • Usalo se preferisci un throughput elevato rispetto a una bassa latenza • Usalo se preferisci i client SDK asincroni	• Non supporta client di servizio che richiedono il supporto HTTP/2 come e `KinesisAsynClient` `TranscribeStreamingAsyncClient` Le seguenti proprietà del sistema Java non sono supportate: • javax.net.ssl.keystore • javax.net.ssl.keyStorePassword • javax.net.ssl.truststore • javax.net.ssl.trustStorePassword

[1] A causa dei loro vantaggi aggiuntivi, si consiglia di utilizzare i client HTTP AWS basati su CRT, se possibile.

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 and Tools Reference Guide.AWS SDKs

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'`auto`utilizzo di una proprietà di sistema impostata in Java.

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

L'esempio seguente mostra come impostare la modalità predefinita sull'`auto`utilizzo di un'-Dopzione 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'`auto`utilizzo di una variabile di ambiente.

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

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 URLConnection basato”

Per visualizzare il set completo di opzioni di configurazione disponibili per`ApacheHttpClient`, consulta ApacheHttpClient.Builder e .Builder. ProxyConfiguration

Accedi a **`ApacheHttpClient`**

Nella maggior parte dei casi, si utilizza `ApacheHttpClient` senza alcuna configurazione esplicita. Dichiari 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 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>2.27.21</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>2.27.21</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 with and values. `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'**`ApacheHttpClient`**istanza

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'`ApacheHttpClient`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 Apache utilizzato da due client di servizio. L'`ApacheHttpClient`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

```
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 URLConnection basato

AWS SDK for Java 2.x Offre un client `UrlConnectionHttpClient` HTTP più leggero rispetto a quello predefinito. `ApacheHttpClient` `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 puoi 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'BatchUpdateFindingsoperazione
> - Tutte le `Update*`operazioni API di Amazon API Gateway
> - Diverse `Update*` operazioni nell' WorkDocsAPI 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`, dichiari 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>2.27.21</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 URLConnection basato.

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

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 URLConnection basato che viene 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 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 URLConnection basato 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. Dichiari 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 richiama 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>2.27.21</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>2.27.21</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'**`NettyNioAsyncHttpClient`**istanza a un client di servizio

Se devi configurare un'istanza di`NettyNioAsyncHttpClient`, ti consigliamo di creare un'`NettyNioAsyncHttpClient`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'`NettyNioAsyncHttpClient`istanza non viene chiusa quando non è più necessaria.

L'esempio seguente crea un'`DynamoDbAsyncClient`istanza che viene utilizzata da un'istanza. `DynamoDbEnhancedAsyncClient` L'`DynamoDbAsyncClient`istanza contiene l'`NettyNioAsyncHttpClient`istanza 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.
```

Configura la negoziazione del protocollo ALPN

ALPN (Application-Layer Protocol Negotiation) è un'estensione TLS che consente al livello dell'applicazione di negoziare quale protocollo deve essere eseguito su una connessione sicura in modo da evitare ulteriori round trip e offrire prestazioni migliori.

Per consentire al client HTTP basato su Netty di utilizzare ALPN, chiamate i metodi builder come illustrato nel seguente frammento:

```
import software.amazon.awssdk.http.Protocol;
import software.amazon.awssdk.http.ProtocolNegotiation;
import software.amazon.awssdk.http.async.SdkAsyncHttpClient;
import software.amazon.awssdk.http.nio.netty.NettyNioAsyncHttpClient;
import
 software.amazon.awssdk.services.transcribestreaming.TranscribeStreamingAsyncClient;


// Configure the Netty-based HTTP client to use the ALPN protocol.
SdkAsyncHttpClient nettyClient = NettyNioAsyncHttpClient.builder()
                                                        .protocol(Protocol.HTTP2)

 .protocolNegotiation(ProtocolNegotiation.ALPN)
                                                        .build();
// Use the Netty-based HTTP client with a service client.
TranscribeStreamingAsyncClient transcribeClient =
 TranscribeStreamingAsyncClient.builder()

 .httpClient(nettyClient)
```

```
  .build();
```

La negoziazione del protocollo ALPN attualmente funziona solo con il protocollo HTTP/2, come mostrato nello snippet precedente.

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. AwsCrtHttpClientAwsCrtAsyncHttpClient I client HTTP AWS basati su CRT offrono i seguenti vantaggi del client HTTP:

- Tempo di avvio dell'SDK più rapido
- Minore ingombro di memoria
- 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 multiparte 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.

Usa una delle seguenti opzioni per configurare il tuo file Maven. `pom.xml`

> **Note**
>
> È possibile scegliere di utilizzare l'opzione jar specifica della piattaforma se è necessario ridurre le dimensioni delle dipendenze di runtime, ad esempio se l'applicazione viene eseguita in una funzione. AWS Lambda

Uber-jar option

Per impostazione predefinita, `aws-crt-client` utilizza un uber-jar di artefatti AWS CRT che contiene file binari per diverse piattaforme, tra cui Linux, Windows e macOS.

```
<project>
   <properties>
     <aws.sdk.java.version>2.29.10*</aws.sdk.java.version>
  </properties>
  <dependencyManagement>
   <dependencies>
      <dependency>
        <groupId>software.amazon.awssdk</groupId>
        <artifactId>bom</artifactId>
        <version>${aws.sdk.version}</version>
        <type>pom</type>
        <scope>import</scope>
      </dependency>
   </dependencies>
  </dependencyManagement>
  <dependencies>
   <dependency>
      <groupId>software.amazon.awssdk</groupId>
      <artifactId>aws-crt-client</artifactId>
   </dependency>
```

```
  </dependencies>
</project>
```

*Sostituisci la versione mostrata in rosso con la versione di Java SDK che desideri utilizzare. Trova le ultime novità su Maven Central.

Platform-specific jar option

Per limitare il runtime Java alla versione specifica della piattaforma della libreria AWS CRT, apporta le seguenti modifiche all'opzione Uber-JAR.

- Aggiungi un `exclusions` elemento all'artefatto dell'SDK. `aws-crt-client` Questa esclusione impedisce all'SDK di utilizzare transitivamente il CRT uber-jar. AWS
- Aggiungi un elemento di dipendenza per la versione specifica della piattaforma CRT di cui hai bisogno. AWS Consultate i passaggi per determinare la versione dell'artefatto AWS CRT di seguito per sapere come determinare la versione corretta.

```
<project>
   <properties>
     <aws.sdk.java.version>2.29.10[1]</aws.sdk.java.version>
  </properties>
   <dependencyManagement>
        <dependencies>
            <dependency>
                <groupId>software.amazon.awssdk</groupId>
                <artifactId>bom</artifactId>
                <version>${aws.sdk.java.version}</version>
                <type>pom</type>
                <scope>import</scope>
            </dependency>
        </dependencies>
    </dependencyManagement>
    <dependencies>
        <dependency>
            <groupId>software.amazon.awssdk</groupId>
            <artifactId>aws-crt-client</artifactId>
            <exclusions>
                <exclusion>
                    <groupId>software.amazon.awssdk.crt</groupId>
                    <artifactId>aws-crt</artifactId>
                </exclusion>
            </exclusions>
```

```
        </dependency>
        <dependency>
            <groupId>software.amazon.awssdk.crt</groupId>
            <artifactId>aws-crt</artifactId>
            <version>0.31.3[2]</version>
            <classifier>linux-x86_64[3]</classifier>
        </dependency>
    </dependencies>
```

[1] Sostituisci la versione mostrata in rosso con la versione di Java SDK che desideri utilizzare. Trova le ultime novità su Maven Central.

[2] Sostituisci la versione `software.amazon.awssdk.crt:aws-crt` che sarebbe fornita dall'opzione Uber-JAR. Vedi i seguenti passaggi per determinare la versione dell'artefatto AWS CRT.

[3] Sostituisci il `classifier` valore con uno per la tua piattaforma. Fate riferimento alla GitHub pagina AWS CRT per Java per un elenco dei valori disponibili.

Passaggi per determinare la versione dell' AWS artefatto CRT

Utilizzate i seguenti passaggi per determinare la versione dell'artefatto AWS CRT compatibile con la versione dell'SDK for Java in uso.

1. Configura il *pom.xml* file come mostrato nell'opzione Uber-JAR. Questa configurazione ti consente di vedere quale versione dell'SDK `software.amazon.awssdk.crt:aws-crt` è disponibile per impostazione predefinita.
2. Alla radice del progetto (nella stessa directory del `pom.xml` file), esegui il seguente comando Maven:

   ```
   mvn dependency:tree -Dincludes=software.amazon.awssdk.crt:aws-crt
   ```

   Maven potrebbe eseguire altre azioni, ma alla fine dovresti vedere l'output della console della `software.amazon.awssdk.crt:aws-crt` dipendenza utilizzata dall'SDK in modo transitivo. Il seguente frammento mostra un output di esempio basato su una versione SDK di: `2.29.10`

   ```
   [INFO] org.example:yourProject:jar:1.0-SNAPSHOT
   [INFO] \- software.amazon.awssdk:aws-crt-client:jar:2.29.10:compile
   [INFO]    \- software.amazon.awssdk.crt:aws-crt:jar:0.31.3:compile
   ```

3. Usa la versione mostrata dalla console per l'artefatto. `software.amazon.awssdk.crt:aws-crt` In questo caso, aggiungilo `0.31.3` al tuo `pom.xml` file.

Usa e configura un client HTTP AWS 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>
         <exclusion>
            <groupId>software.amazon.awssdk</groupId>
            <artifactId>apache-client</artifactId>
         </exclusion>
      </exclusions>
   </dependency>
   <dependency>
      <groupId>software.amazon.awssdk</groupId>
      <artifactId>aws-crt-client</artifactId>
   </dependency>
```

```
  </dependencies>
</project>
```

Visita il repository centrale di Maven per le ultime novità. *VERSION*

> **Note**
>
> Se in un `pom.xml` file vengono dichiarati più client di servizio, tutti richiedono l'`exclusions`elemento 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.AwsCrtSdkHttpService`

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.AwsCrtSdkHttpService
```

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.AwsCrtSdkHttpService");
```

> **Note**
>
> È necessario aggiungere una dipendenza dall'elemento nel `pom.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. TranscribeStreamingAsyncClientNettyNioAsyncHttpClient

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
- URLConnectionclient HTTP basato su
- 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

Proprietà del sistema	Descrizione	Supporto client HTTP
http.proxyUser	Nome utente per l'autenti cazione del proxy HTTP	Tutti
Http.ProxyPassword	Password per l'autenticazione del proxy HTTP	Tutti
http. nonProxyHosts	Elenco di host che devono essere raggiunti direttamente, ignorando il proxy. Questo elenco è valido anche quando si utilizza HTTPS.	Tutti
https://proxyHost	Nome host del server proxy HTTPS	Netty, CRT
Porta https://proxy	Numero di porta del server proxy HTTPS	Netty, CRT
Utente https://proxy	Nome utente per l'autenti cazione 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 1	Un URL valido con uno schema HTTP	Tutti
HTTPS_PROXY 1	Un URL valido con uno schema HTTPS	Netty, CRT
NO_PROXY 2	Elenco di host che devono essere raggiunti direttamente, ignorando il proxy. L'elenco è	Tutti

Variabile di ambiente	Descrizione	Supporto client HTTP
	valido sia per HTTP che per HTTPS.	

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, (e). `AwsCrtHttpClient` `AwsCrtAsyncHttpClient`

[1] La variabile di ambiente richiesta, indipendentemente dal fatto che sia `HTTP_PROXY` o menoHTTPS_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();
```

[2] 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 vengono AWS SDK for Java 2.x generate le eccezioni è importante per creare 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 sono controllate?

AWS SDK per Java Utilizza eccezioni di runtime (o non controllate) anziché eccezioni controllate per questi 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)

AwsServiceExceptionè l'eccezione più comune che si verificherà quando si utilizza. AWS SDK per 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 AwsErrorDetailsclasse
- AWS ID 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`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

SdkClientExceptionindica 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 per Java genera una connessione di rete `SdkClientException` se non è disponibile alcuna connessione di rete quando si tenta di richiamare un'operazione su uno dei client.

Eccezioni e comportamento di ripetizione dei 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`.

Tentativi

Le chiamate a Servizi AWS possono fallire occasionalmente per motivi imprevisti. Alcuni errori, come la limitazione (frequenza superata) o gli errori transitori, potrebbero avere esito positivo se la chiamata viene ritentata. AWS SDK for Java 2.x Dispone di un meccanismo integrato per rilevare tali errori e riprovare automaticamente la chiamata, abilitato di default per tutti i client.

Questa pagina descrive come funziona, come configurare le diverse modalità e personalizzare il comportamento dei tentativi.

Strategie di nuovo tentativo

Una strategia di nuovo tentativo è un meccanismo utilizzato nell'SDK per implementare nuovi tentativi. Ogni client SDK dispone di una strategia di riprova creata in fase di compilazione che non può essere modificata dopo la creazione del client.

La strategia di riprova ha le seguenti responsabilità.

- Classificate le eccezioni come riprovabili o meno.
- Calcola il ritardo suggerito per attendere prima del tentativo successivo.
- Mantieni un token bucket che fornisca un meccanismo per bloccare i nuovi tentativi quando una grande percentuale di richieste ha esito negativo e i nuovi tentativi non hanno esito positivo.

> **Note**
>
> Prima del rilascio delle strategie di riprova con la versione 2.26.0 dell'SDK, le politiche di riprova fornivano il meccanismo di riprova dell'SDK. L'API retry policy è costituita dalla *RetryPolicy*classe principale del *software.amazon.awssdk.core.retry* pacchetto, mentre il pacchetto contiene gli elementi dell'API retry strategy. *software.amazon.awssdk.retries*
> L'API retry strategy è stata introdotta come parte dell' AWS ampio sforzo volto a unificare le interfacce e il comportamento dei componenti principali di. SDKs

L'SDK for Java 2.x include tre strategie di riprova integrate: standard, legacy e adattiva. Tutte e tre le strategie di riprova sono preconfigurate per riprovare una serie di eccezioni riprovabili. Esempi di errori riutilizzabili sono i timeout dei socket, la limitazione sul lato del servizio, gli errori di blocco simultanei o ottimistici e gli errori di servizio transitori.

Strategia di riprova standard

La strategia di riprova standard è l'`RetryStrategy`implementazione consigliata per i normali casi d'uso. A differenza della`AdaptiveRetryStrategy`, la strategia standard è generalmente utile in tutti i casi d'uso con nuovi tentativi.

Per impostazione predefinita, la strategia di ripetizione standard esegue le seguenti operazioni.

- Riprova in base alle condizioni configurate in fase di compilazione. Puoi regolarlo con`StandardRetryStrategy.Builder`#`retryOnException`.
- Riprova 2 volte per un totale di 3 tentativi. Puoi regolarlo con`StandardRetryStrategy.Builder#maxAttempts(int)`.
- Per le eccezioni che non prevedono il throttling, utilizza la strategia di `BackoffStrategy`#`exponentialDelay` backoff, con un ritardo base di 100 millisecondi e un ritardo massimo di 20 secondi. Puoi `StandardRetryStrategy.Builder#backoffStrategy` regolarlo con.
- Per le eccezioni relative alla limitazione, utilizza la strategia di `BackoffStrategy#exponentialDelay` backoff, con un ritardo base di 1 secondo e un ritardo massimo di 20 secondi. Puoi regolarlo con. `StandardRetryStrategy.Builder#throttlingBackoffStrategy`

- Interrompe il circuito (disabilita i nuovi tentativi) in caso di gravi guasti a valle. Il primo tentativo viene sempre eseguito, solo i nuovi tentativi sono disabilitati. Regola con`StandardRetryStrategy.Builder#circuitBreakerEnabled`.

Strategia preesistente di nuovi tentativi

La precedente strategia di ripetizione dei tentativi è `RetryStrategy` destinata ai casi d'uso normali, tuttavia è obsoleta a favore di. `StandardRetryStrategy` Questa è la strategia di riprova predefinita utilizzata dai clienti quando non si specifica un'altra strategia.

È caratterizzata dal fatto che le eccezioni di throttling e non throttling vengono trattate in modo diverso, per le eccezioni di throttling il ritardo di base per il backoff è maggiore (500 ms) rispetto al ritardo di base per le eccezioni senza limitazione (100 ms) e le eccezioni di throttling non influiscono sullo stato del token bucket.

L'esperienza nell'uso di questa strategia su larga scala ha dimostrato che non è particolarmente migliore della strategia di riprova standard. AWS Inoltre, non riesce a proteggere i servizi a valle dalle tempeste di tentativi ripetuti e può portare alla carenza di risorse da parte del client.

Per impostazione predefinita, la strategia di ripetizione dei tentativi precedente esegue le seguenti operazioni.

- Riprova in base alle condizioni configurate in fase di compilazione. Puoi regolarlo con`LegacyRetryStrategy.Builder`#`retryOnException`.
- Riprova 3 volte per un totale di 4 tentativi. Puoi regolarlo con`LegacyRetryStrategy.Builder#maxAttempts(int)`.
- Per le eccezioni che non prevedono il throttling, utilizza la strategia di `BackoffStrategy#exponentialDelay` backoff, con un ritardo base di 100 millisecondi e un ritardo massimo di 20 secondi. Puoi regolarlo con `LegacyRetryStrategy.Builder#backoffStrategy.`
- Per le eccezioni di limitazione, utilizza la strategia di `BackoffStrategy#exponentialDelay` backoff, con un ritardo base di 500 millisecondi e un ritardo massimo di 20 secondi. Puoi regolarlo `LegacyRetryStrategy.Builder#throttlingBackoffStrategy` con.
- Interrompe il circuito (disabilita i nuovi tentativi) in caso di gravi guasti a valle. L'interruzione del circuito non impedisce mai il successo del primo tentativo. È possibile modificare questo comportamento con`LegacyRetryStrategy.Builder#circuitBreakerEnabled`.
- Lo stato dell'interruttore automatico non è influenzato dalle eccezioni di limitazione.

Strategia adattiva di nuovi tentativi

La strategia adattiva di riprova è `RetryStrategy` adatta ai casi d'uso con un elevato livello di vincoli di risorse.

La strategia adattiva di riprova include tutte le funzionalità della strategia standard e aggiunge un limitatore di velocità sul lato client che misura la frequenza delle richieste limitate rispetto alle richieste non limitate. La strategia utilizza questa misurazione per rallentare le richieste nel tentativo di rimanere entro una larghezza di banda sicura, causando idealmente zero errori di limitazione.

Per impostazione predefinita, la strategia di ripetizione adattiva esegue le seguenti operazioni.

- Riprova in base alle condizioni configurate in fase di compilazione. Puoi regolarlo con`AdaptiveRetryStrategy.Builder#retryOnException`.
- Riprova 2 volte per un totale di 3 tentativi. Puoi regolarlo con`AdaptiveRetryStrategy.Builder#maxAttempts(int)`.
- Utilizza un ritardo di backoff dinamico basato sul carico corrente rispetto alla risorsa a valle.
- Effettua l'interruzione del circuito (disabilitando i nuovi tentativi) in caso di un numero elevato di guasti a valle. L'interruzione del circuito può impedire un secondo tentativo in scenari di interruzione per proteggere il servizio a valle.

> **Warning**
>
> La strategia di ripetizione adattiva presuppone che il client lavori su una singola risorsa (ad esempio, una tabella DynamoDB o un bucket Amazon S3).
> Se si utilizza un singolo client per più risorse, le limitazioni o le interruzioni associate a una risorsa comportano un aumento della latenza e degli errori quando il client accede a tutte le altre risorse. Quando si utilizza la strategia di ripetizione adattiva, si consiglia di utilizzare un singolo client per ogni risorsa.
> Si consiglia inoltre di utilizzare questa strategia in situazioni in cui tutti i client utilizzano la strategia adattiva di ripetizione dei tentativi rispetto alla risorsa.

> **Important**
>
> Il rilascio delle strategie di ripetizione con la versione 2.26.0 di Java SDK include il nuovo valore di enumerazione. `RetryMode.ADAPTIVE_V2` La `ADAPTIVE_V2` modalità corregge un

errore che non è riuscito a ritardare il primo tentativo quando in precedenza erano stati rilevati errori di limitazione.
Con la versione 2.26.0, gli utenti ottengono automaticamente il comportamento della `ADAPTIVE_V2` modalità impostando la modalità come `adaptive` con una variabile di ambiente, una proprietà di sistema o un'impostazione del profilo. Non esiste alcun `adaptive_v2` valore per queste impostazioni. Per informazioni su come impostare la modalità, vedere la the section called “Specificare una strategia” sezione seguente.
Gli utenti possono ottenere il comportamento precedente impostando la modalità nel codice utilizzando`RetryMode.ADAPTIVE`.

Riepilogo: confronto dei valori predefiniti della strategia Retry

La tabella seguente mostra i valori predefiniti per le proprietà di ogni strategia di nuovo tentativo.

Strategia	Numero massimo di tentativi	Ritardo di base per errori che non riguardano la limitazione	Ritardo di base per errori di limitazione	Dimensioni del bucket token	Costo del token per ogni nuovo tentativo senza limitazione	Costo del token per ogni nuovo tentativo di limitazione
Standard	3	100 ms	1000 ms	500	5	5
Legacy	4	100 ms	500 ms	500	5	0
Adattabile	3	100 ms	100 ms	500	5	5

Specificare una strategia

Avete quattro modi per specificare una strategia per il vostro cliente di assistenza.

In codice

Quando crei un client, puoi configurare un'espressione lambda con una strategia di riprova. Il seguente frammento configura una strategia di riprova standard che utilizza valori predefiniti su un client di servizio DynamoDB.

```
DynamoDbClient client = DynamoDbClient.builder()
        .overrideConfiguration(o -> o.retryStrategy(RetryMode.STANDARD))
        .build();
```

È possibile specificare o al posto di. `RetryMode.LEGACY` `RetryMode.ADAPTIVE` `RetryMode.STANDARD`

Come impostazione del profilo

Includi `retry_mode` come impostazione del profilo nel file di AWS configurazione condiviso. `legacy`Specificare `standard` o `adaptive` come valore. Se impostata come impostazione di profilo, tutti i client di servizio creati mentre il profilo è attivo utilizzano la strategia di ripetizione dei tentativi specificata con valori predefiniti. È possibile ignorare questa impostazione configurando una strategia di nuovo tentativo nel codice, come mostrato in precedenza.

Con il seguente profilo, tutti i client di servizio utilizzano la strategia di riprova standard.

```
[profile dev]
region = us-east-2
retry_mode = standard
```

Come proprietà del sistema JVM

È possibile configurare una strategia di riprova per tutti i client di servizio, a meno che non sia sovrascritta nel codice, utilizzando la proprietà di sistema. `aws.retryMode` Specificare `standard` o come `legacy` valore. `adaptive`

Utilizzate lo `-D` switch quando richiamate Java, come illustrato nel comando seguente.

```
java -Daws.retryMode=standard ...
```

In alternativa, impostate la proprietà di sistema nel codice prima di creare qualsiasi client, come mostrato nel frammento seguente.

```
public void main(String[] args) {
    // Set the property BEFORE any AWS service clients are created.
    System.setProperty("aws.retryMode", "standard");
    ...
}
```

Con una variabile di ambiente

È inoltre possibile utilizzare la variabile di `AWS_RETRY_MODE` ambiente con un valore di `standard`legacy, `adaptive`. Analogamente all'impostazione del profilo o alla proprietà del sistema JVM, la variabile di ambiente configura tutti i client di servizio con la modalità di riprova specificata, a meno che non si configuri un client nel codice.

Il comando seguente imposta la modalità retry `standard` per la sessione di shell corrente.

```
export AWS_RETRY_MODE=standard
```

Personalizza una strategia

Puoi personalizzare qualsiasi strategia di nuovo tentativo impostando il numero massimo di tentativi, la strategia di backoff e le eccezioni che possono essere riprovati. È possibile personalizzare quando si crea una strategia di ripetizione dei tentativi o quando si crea un client utilizzando un generatore di override che consente ulteriori perfezionamenti della strategia configurata.

Personalizza il numero massimo di tentativi

È possibile configurare il numero massimo di tentativi durante la costruzione del client, come illustrato nella seguente dichiarazione. L'istruzione seguente personalizza la strategia di ripetizione dei tentativi predefinita per il client fino a un massimo di 5 tentativi, un primo tentativo più 4 tentativi.

```
DynamoDbClient client = DynamoDbClient.builder()
        .overrideConfiguration(o -> o.retryStrategy(b -> b.maxAttempts(5)))
        .build();
```

In alternativa, è possibile creare la strategia e fornirla al client come illustrato nel seguente esempio di codice. Il codice seguente sostituisce i 3 tentativi massimi standard con 10 e configura un client DynamoDB con la strategia personalizzata.

```
StandardRetryStrategy strategy = AwsRetryStrategy.standardRetryStrategy()
         .toBuilder()
         .maxAttempts(10)
         .build();
DynamoDbClient client = DynamoDbClient.builder()
        .overrideConfiguration(o -> o.retryStrategy(strategy))
        .build();
```

> **Warning**
>
> Si consiglia di configurare ogni client con un'istanza unica. `RetryStrategy` Se un'`RetryStrategy`istanza è condivisa, gli errori in un client potrebbero influire sul comportamento dei tentativi nell'altro.

È inoltre possibile impostare il numero massimo di tentativi per tutti i client utilizzando impostazioni esterne anziché codice. È possibile configurare questa impostazione come descritto nella the section called “Specificare una strategia” sezione.

Personalizza le eccezioni riprovabili

È possibile configurare eccezioni aggiuntive che attivano nuovi tentativi durante la creazione del client. Questa personalizzazione viene fornita per i casi limite in cui vengono generate eccezioni che non sono incluse nel set predefinito di eccezioni riutilizzabili.

I `retryOnExceptionOrCause` metodi `retryOnException` and aggiungono nuovi tipi di eccezione al set esistente di eccezioni riutilizzabili; non sostituiscono il set predefinito. Ciò consente di estendere il comportamento dei tentativi mantenendo le funzionalità di ripetizione predefinite dell'SDK.

Il `retryOnExceptionOrCause` metodo aggiunge un'eccezione riutilizzabile se l'SDK genera l'eccezione diretta o se l'eccezione viene inserita come causa in un'altra eccezione.

Il seguente frammento di codice mostra i metodi utilizzati per personalizzare le eccezioni relative ai tentativi: e. `retryOnException` `retryOnExceptionOrCause` I `retryOnExceptionOrCause` metodi aggiungono un'eccezione riutilizzabile se l'SDK genera l'eccezione diretta o se l'eccezione è inclusa.

```
DynamoDbClient client = DynamoDbClient.builder()
        .overrideConfiguration(o -> o.retryStrategy(
                b -> b.retryOnException(EdgeCaseException.class)
                      .retryOnExceptionOrCause(WrappedEdgeCaseException.class)))
        .build();
```

Personalizza la strategia di backoff

È possibile creare la strategia di backoff e fornirla al cliente.

Il codice seguente ne crea una `BackoffStrategy` che sostituisce la strategia di backoff esponenziale con ritardo esponenziale predefinita della strategia standard.

```
BackoffStrategy backoffStrategy =
        BackoffStrategy.exponentialDelay(Duration.ofMillis(150),  // The base delay.
                                         Duration.ofSeconds(15)); // The maximum delay.
DynamoDbClient client = DynamoDbClient.builder()
        .overrideConfiguration(o -> o.retryStrategy(
                b -> b.backoffStrategy(backoffStrategy)))
        .build();
```

Migrazione da **`RetryPolicy`** a **`RetryStrategy`**

`RetryPolicy`(l'API retry policy) sarà supportata per il prossimo futuro. Se attualmente utilizzi un'istanza di `RetryPolicy` per configurare il tuo client, tutto funzionerà come prima. Dietro le quinte, Java SDK lo adatta a un. `RetryStrategy` Le nuove interfacce retry strategy offrono le stesse funzionalità di un `RetryPolicy` ma vengono create e configurate in modo diverso.

Implementare **`ContentStreamProvider`** nel AWS SDK for Java 2.x

`ContentStreamProvider`è un'astrazione utilizzata in per consentire letture multiple dei AWS SDK for Java 2.x dati di input. Questo argomento spiega come implementarlo `ContentStreamProvider` correttamente per le applicazioni.

L'SDK for Java 2.x utilizza `ContentStreamProvider#newStream()` il metodo ogni volta che deve leggere un intero flusso. Affinché ciò funzioni per l'intero flusso, lo stream restituito deve essere sempre all'inizio del contenuto e deve contenere gli stessi dati.

Nelle sezioni seguenti, forniamo tre approcci per implementare correttamente questo comportamento.

Uso di **`mark()`** e **`reset()`**

Nell'esempio seguente, utilizziamo `mark(int)` il costruttore prima che inizi la lettura per assicurarci di poter ripristinare lo stream all'inizio. Per ogni invocazione di `newStream()` resettiamo lo stream:

```
public class MyContentStreamProvider implements ContentStreamProvider {
    private InputStream contentStream;
```

```
    public MyContentStreamProvider(InputStream contentStream) {
        this.contentStream = contentStream;
        this.contentStream.mark(MAX_LEN);
    }

    @Override
    public InputStream newStream() {
        contentStream.reset();
        return contentStream;
    }
}
```

Usa il buffering se **`mark()`** e **`reset()`** non sono disponibili

Se il tuo stream non supporta `reset()` direttamente `mark()` e direttamente, puoi comunque utilizzare la soluzione mostrata in precedenza, racchiudendo prima lo stream in: `BufferedInputStream`

```
public class MyContentStreamProvider implements ContentStreamProvider {
    private BufferedReader contentStream;

    public MyContentStreamProvider(InputStream contentStream) {
        this.contentStream = new BufferedInputStream(contentStream);
        this.contentStream.mark(MAX_LEN);
    }

    @Override
    public InputStream newStream() {
        contentStream.reset();
        return contentStream;
    }
}
```

Crea nuovi stream

Un approccio più semplice consiste semplicemente nell'ottenere un nuovo flusso di dati a ogni chiamata e chiudere quello precedente:

```
public class MyContentStreamProvider implements ContentStreamProvider {
    private InputStream contentStream;
```

```
    @Override
    public InputStream newStream() {
        if (contentStream != null) {
            contentStream.close();
        }
        contentStream = openStream();
        return contentStream;
    }
}
```

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, tra gli altri, Java Logging Framework e Apache Log4j 2. 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` Log4j 2. 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 che si utilizza con è. AWS SDK for Java 2.x `software.amazon.awssdk`

Aggiungi dipendenza dalla registrazione

Per configurare l'associazione Log4j 2 per SLF4 J nel tuo file di build, usa quanto segue.

Maven

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

```
...
<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 IDs le richieste sono accessibili a livello di codice tramite `SdkServiceException`oggetti nell'SDK per qualsiasi chiamata di servizio non riuscita e possono anche essere segnalate 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:
 QS9DUMME2NHEDH8TGT9N5V53OJVV4KQNSO5AEMVJF66Q9ASUAAJG, 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 restituisce 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: DefaultSdkHttpFullRequest(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
ApacheHttpClient (predefinito)	NettyNioAsyncHttpClient (predefinito)
UrlConnectionHttpClient	AwsCrtAsyncHttpClient
AwsCrtHttpClient	

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=e367fa0bc217a6a65675bb743e1280cf12fbe8d566196a816d948fdf0b42ca1a}{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: sAFeZDOKdUMsBbkdjyDZw7P0oocb4C9KbiuzfJ6TWKQsGXHM/
dFuOvr2tUb7Y1wEHGdJ3DSIxq0=}{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/AwsCrtHttpClient

Se il client di servizio è stato configurato per utilizzare un'istanza di un client HTTP AWS basato su CRT, 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 di sistema:

```
-Daws.crt.log.leve
l=Trace
-Daws.crt.log.des
tination=File
-Daws.crt.log.fil
ename=<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.leve
l=Trace
```

A livello di codice:

```
import software.amazon.awssdk.crt.Log;

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

`-Daws.crt.log.destination=Stdout`	

Per motivi di sicurezza, a livello «Trace» i client HTTP AWS basati su CRT registrano solo le intestazioni di risposta. 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.

Le migliori pratiche per AWS SDK for Java 2.x

Riutilizza i client di servizio, se possibile

Ogni client di servizio 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 di servizio sono thread-safe.

Se non vuoi condividere un'istanza client, chiama `close()` l'istanza per rilasciare le risorse quando il client non è necessario.

Chiudi i client di servizio, se non è più necessario

Chiudi un client di servizio per rilasciare risorse, come i thread, se non sono più necessarie. 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.

Risoluzione dei problemi FAQs

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 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 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 la richiesta IDs e l'ora 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 intermediaria (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.

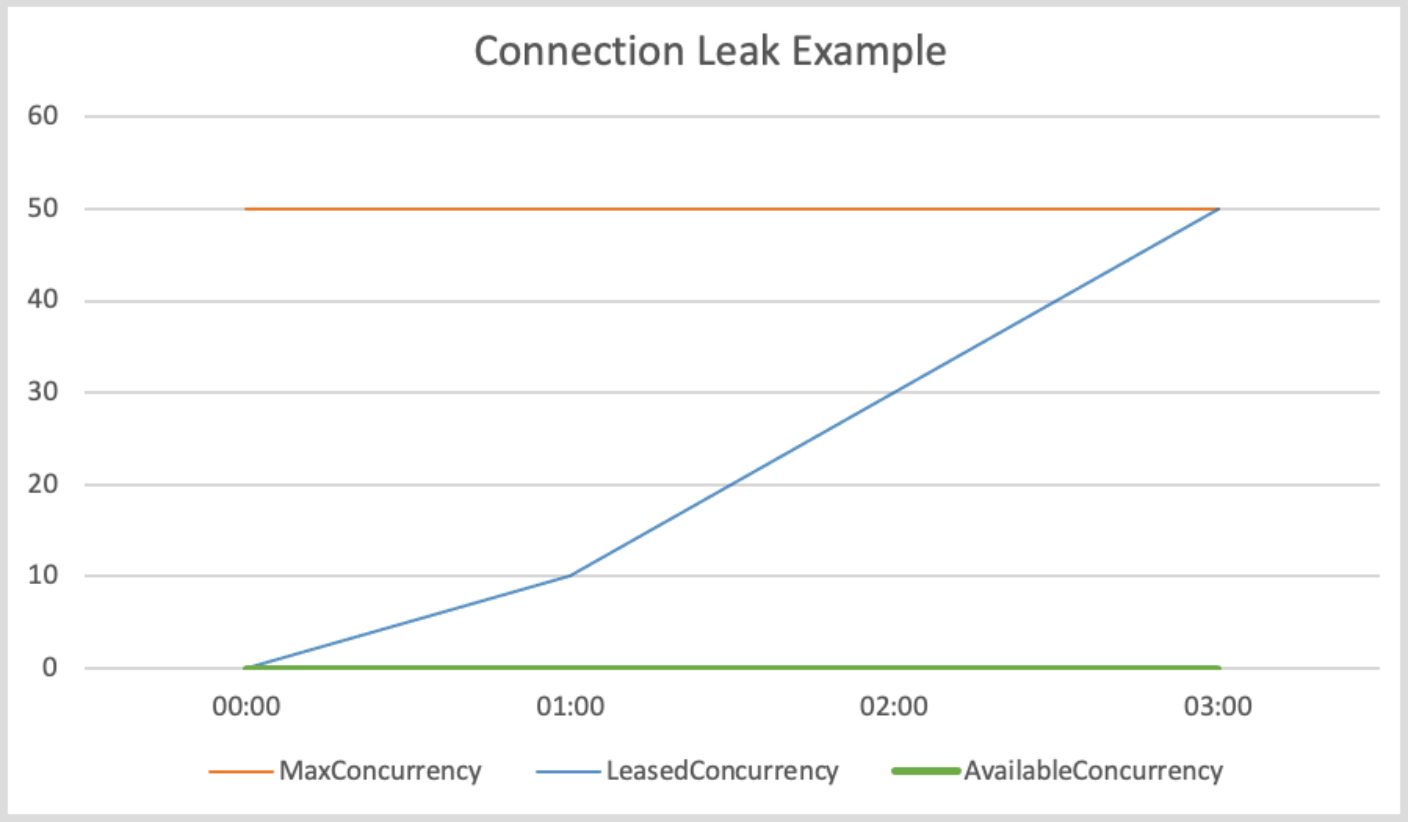
Prova a eseguire un dump TCP per una connessione interessata nella periferia client della rete (ad esempio, dopo tutti i proxy che controlli).

Se vedi che l' AWS endpoint sta inviando un `TCP RST` (reset), contatta il servizio interessato. Preparati a fornire la richiesta IDs e l'ora 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.

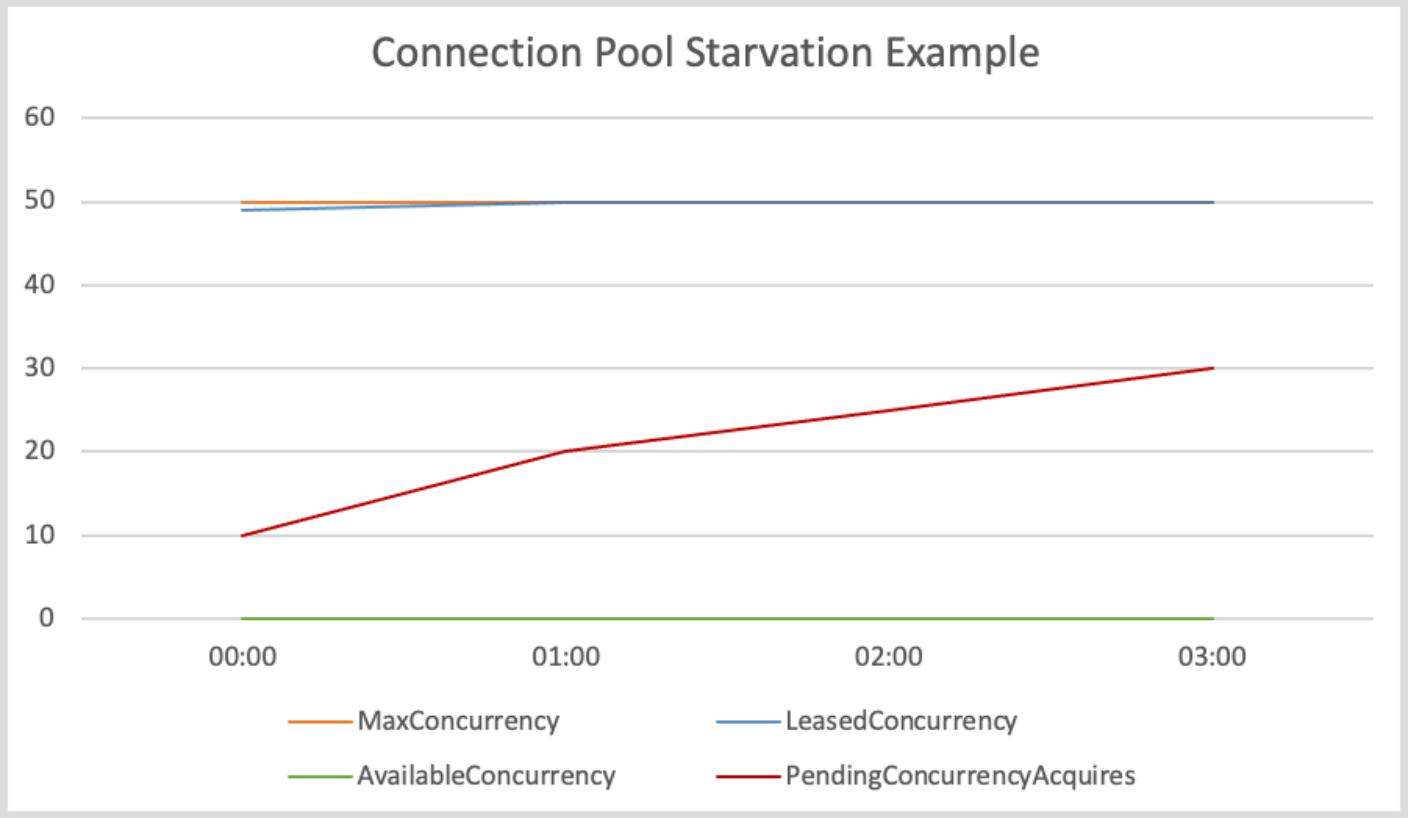
Come posso correggere l'errore «Impossibile eseguire la richiesta HTTP: timeout in attesa di connessione dal pool»?

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 `LeasedConcurrency`, `AvailableConcurrency`, e `MaxConcurrency` metriche. Se `LeasedConcurrency` aumenta fino a raggiungere l'`MaxConcurrency`obiettivo 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.

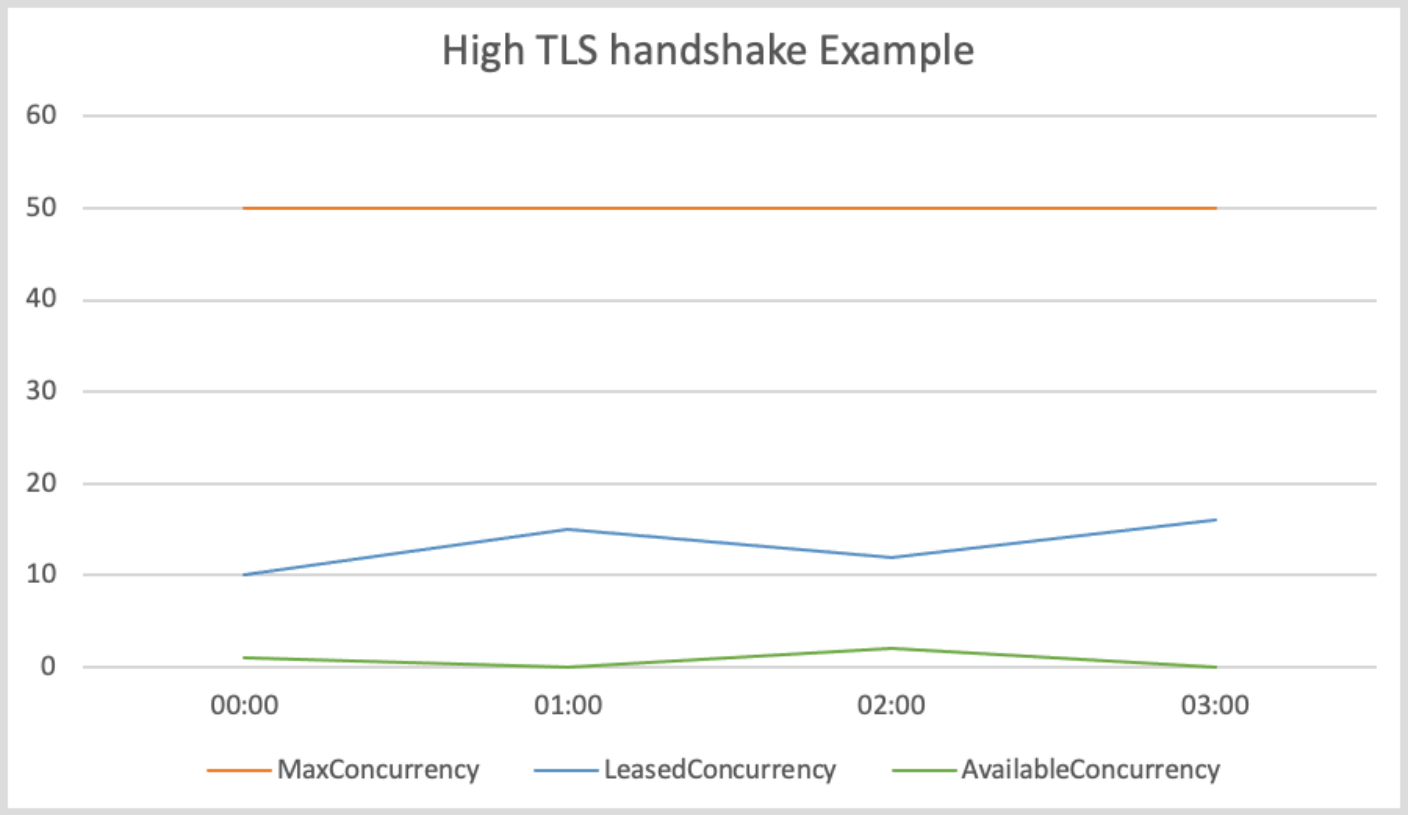
- Aumentare la dimensione del pool di connessioni,
- Aumentare il timeout di acquisizione.
- Rallenta la frequenza delle richieste.

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'`NettyNioAsyncHttpClient`istanza 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? **NoSuchMethodErrorNoSuchFieldError**

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 inizializzatore 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 per 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 per 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 questa opzione 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` s o `Error` s 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`

AWS SDK per Java Utilizza le funzionalità della versione 2.x

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 per 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 per Java versione 1.0, la risposta contiene un token da utilizzare per recuperare la pagina successiva dei risultati. Al contrario, la versione AWS

SDK per 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.paginators.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 per Java 2.x: Waiters

L'utilità waiters della versione AWS SDK per 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 per Java 2.x. AWS SDK per 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 per Java

Per esempio:

```
<project>
  <dependencyManagement>
   <dependencies>
      <dependency>
        <groupId>software.amazon.awssdk</groupId>
        <artifactId>bom</artifactId>
        <version>2.27.21</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.waitUntilTableNotExists(b -> b.tableName("myTable"),
```

```
                o -> o.maxAttempts(10));

asyncWaiter.waitUntilTableExists(b -> b.tableName("myTable"),
                  o -> o.waitTimeout(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, riassegnando 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 per Java CompletableFuture restituiti che consentono di accedere alla risposta quando è pronta.

Usa un client asincrono APIs

Le firme dei metodi client asincroni sono le stesse della loro controparte sincrona, ma i metodi asincroni restituiscono un CompletableFutureoggetto che contiene i risultati dell'operazione asincrona futura. Se viene generato un errore durante l'esecuzione del metodo asincrono dell'SDK, l'errore viene generato come. `CompletionException`

Un approccio che puoi utilizzare per ottenere il risultato consiste nel concatenare un `whenComplete()` metodo alla chiamata al metodo SDK restituita`CompletableFuture`. Il `whenComplete()` metodo riceve il risultato o un oggetto Throwable di tipo `CompletionException`

a seconda del completamento della chiamata asincrona. Si esegue un'azione per elaborare o controllare `whenComplete()` i risultati prima che vengano restituiti al codice chiamante.

Se vuoi restituire qualcosa di diverso dall'oggetto restituito dal metodo SDK, usa invece il `handle()` metodo. Il `handle()` metodo accetta gli stessi parametri`whenComplete()`, ma potete elaborare il risultato e restituire un oggetto.

Per attendere il completamento della catena asincrona e recuperare i risultati del completamento, potete chiamare il metodo. `join()` Se l'`Throwable`oggetto non è stato gestito nella catena, il `join()` metodo genera un valore `CompletionException` deselezionato che racchiude l'eccezione originale. Si accede all'eccezione originale con. `CompletionException#getCause()` Puoi anche chiamare il `CompletableFuture#get()` metodo per ottenere i risultati del completamento. Il `get()` metodo, tuttavia, può generare eccezioni verificate.

L'esempio seguente mostra due varianti di come lavorare con il `listTables()` metodo del client asincrono DynamoDB. L'azione passata a registra `whenComplete()` semplicemente una risposta riuscita, mentre la `handle()` versione estrae l'elenco dei nomi delle tabelle e restituisce l'elenco. In entrambi i casi, se viene generato un errore nella catena asincrona, l'errore viene generato nuovamente in modo che il codice client abbia la possibilità di gestirlo.

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

whenComplete() variation

```
public class DynamoDbAsyncListTables {

    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        DynamoDbAsyncClient dynamoDbAsyncClient =
 DynamoDbAsyncClient.builder().region(region).build();
        try {
```

```
            ListTablesResponse listTablesResponse =
listTablesWhenComplete(dynamoDbAsyncClient).join();  // The join() method may throw
a CompletionException.
            if (listTablesResponse.hasTableNames()){
                System.out.println("Table exist in this region: " + region.id());
            }
        } catch (RuntimeException e) {
            // Handle as needed. Here we simply print out the class names.
            System.out.println(e.getClass()); // Prints 'class
java.util.concurrent.CompletionException'.
            System.out.println(e.getCause().getClass()); // Prints 'class
software.amazon.awssdk.services.dynamodb.model.DynamoDbException'.
        }
    }

    public static CompletableFuture<ListTablesResponse>
listTablesWhenComplete(DynamoDbAsyncClient client) {
        return client.listTables(ListTablesRequest.builder().build())
            .whenComplete((listTablesResponse, throwable) -> {
                if (listTablesResponse != null) {  // Consume the response.
                    System.out.println("The SDK's listTables method completed
successfully.");
                } else {
                    RuntimeException cause = (RuntimeException)
throwable.getCause(); // If an error was thrown during the SDK's listTables method
it is wrapped in a CompletionException.

 // The SDK throws only RuntimeExceptions, so this is a safe cast.
                    System.out.println(cause.getMessage());  // Log error here, but
rethrow so the calling code can handle as needed.
                    throw cause;
                }
            });
    }
```

handle() variation

```
public class DynamoDbAsyncListTables {

    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        DynamoDbAsyncClient dynamoDbAsyncClient =
 DynamoDbAsyncClient.builder().region(region).build();
```

```
        try {
            List<String> tableNames =
 listTablesHandle(dynamoDbAsyncClient).join(); // The join() method may throw a
 CompletionException.
            tableNames.forEach(System.out::println);
        } catch (RuntimeException e) {
            // Handle as needed. Here we simply print out the class names.
            System.out.println(e.getClass()); // Prints 'class
 java.util.concurrent.CompletionException'.
            System.out.println(e.getCause().getClass()); // Prints 'class
 software.amazon.awssdk.services.dynamodb.model.DynamoDbException'.
        }
    }

    public static CompletableFuture<List<String>>
 listTablesHandle(DynamoDbAsyncClient client) {
        return client.listTables(ListTablesRequest.builder().build())
            .handle((listTablesResponse, throwable) -> {
                if (listTablesResponse != null) {
                    return listTablesResponse.tableNames(); // Return the list of
 table names.
                } else {
                    RuntimeException cause = (RuntimeException)
 throwable.getCause(); // If an error was thrown during the SDK's listTables method
 it is wrapped in a CompletionException.

  // The SDK throws only RuntimeExceptions, so this is a safe cast.
                    System.out.println(cause.getMessage());  // Log error here, but
 rethrow so the calling code can handle as needed.
                    throw cause;
                }
            });
    }
}
```

Gestisci lo streaming con metodi asincroni

Per i metodi asincroni con contenuti in streaming, è necessario fornire il contenuto in modo incrementale oppure un codice AsyncRequestBodyper ricevere ed elaborare la risposta AsyncResponseTransformer.

L'esempio seguente carica un file in modalità asincrona utilizzando la forma Amazon S3 asincrona dell'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 Amazon S3 da utilizzando la forma asincrona dell'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();
    }
}
```

Configura opzioni asincrone avanzate

La AWS SDK per Java 2.x utilizza Netty, un framework applicativo di rete asincrono basato sugli eventi, per gestire i thread di I/O. Il AWS SDK per 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`

È possibile specificare un'implementazione JDK specifica quando si crea il client asincrono. `ExecutorService` Il seguente frammento crea un file con un `ExecutorService` numero fisso di thread.

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 esecutore del pool di thread e includerlo nella 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 per 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'aggiornamento alla versione più recente SDKs utilizzeranno automaticamente HTTP/2 quando è supportato dal servizio con cui stanno lavorando. 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 per Java 2.x presenta nuove funzionalità APIs per lo streaming di eventi che implementano il protocollo HTTP/2. Per esempi di come utilizzare queste novità APIs, consulta Lavorare con Kinesis.

Pubblica le metriche SDK dal AWS SDK per Java

Con la AWS SDK per Java versione 2.x, puoi raccogliere metriche sui client di servizio e sulle richieste 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 consente di attivarlo e configurarlo.

Quali sono le diverse **`MetricPublisher`** implementazioni?

L'SDK for Java 2.x offre tre implementazioni dell'interfaccia. `MetricPublisher` Ogni implementazione è pensata per diversi casi d'uso e illustrata nella tabella seguente:

MetricPublisher implementazione	Caso d'uso adatto
`CloudWatchMetricPublisher`	Applicazioni a lunga durata
`EmfMetricLoggingPublisher`	AWS Lambda funzioni
`LoggingMetricPublisher`	Uscita da console per la risoluzione dei problemi

Pubblica i parametri SDK per le applicazioni a esecuzione prolungata

Poiché l'`CloudWatchMetricPublisher`implementazione aggrega e carica periodicamente le metriche su Amazon CloudWatch con un certo ritardo, il suo utilizzo è più adatto per applicazioni a lunga durata.

Le impostazioni predefinite dell'editore delle metriche hanno lo scopo di ridurre al minimo l'utilizzo e i CloudWatch costi della memoria, fornendo al contempo un'utile quantità di informazioni sui dati metrici.

Configurazione

Prima di poter abilitare e utilizzare le metriche utilizzando`CloudWatchMetricPublisher`, completa i seguenti passaggi.

Passaggio 1: aggiungere la dipendenza richiesta

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 per Java

Includi l'`cloudwatch-metric-publisher`ArtifactID 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>2.30.11</version>  <!-- Navigate the link to see the latest version.
 -->
        <type>pom</type>
        <scope>import</scope>
      </dependency>
   </dependencies>
  </dependencyManagement>
  <dependencies>
   <dependency>
      <groupId>software.amazon.awssdk</groupId>
      <artifactId>cloudwatch-metric-publisher</artifactId>
```

```
    </dependency>
  </dependencies>
</project>
```

Passaggio 2: configura le autorizzazioni richieste

Abilita `cloudwatch:PutMetricData` le autorizzazioni per l'identità IAM utilizzata dall'editore delle metriche per consentire all'SDK for Java di scrivere metriche.

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 un editore di metriche CloudWatch

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.

Pubblica le metriche SDK per le funzioni AWS Lambda

Poiché le funzioni Lambda vengono in genere eseguite da millisecondi a minuti, qualsiasi ritardo nell'invio delle metriche, che si verifica con il, rischia la `CloudWatchMetricPublisher` perdita di dati.

`EmfMetricLoggingPublisher`fornisce un approccio più adatto scrivendo immediatamente le metriche come voci di registro strutturate in CloudWatch Embedded Metric Format (EMF). `EmfMetricLoggingPublisher`funziona in ambienti di esecuzione dotati di integrazione integrata con Amazon CloudWatch Logs come AWS Lambda Amazon Elastic Container Service.

Configurazione

Prima di poter abilitare e utilizzare le metriche utilizzando`EmfMetricLoggingPublisher`, completa i seguenti passaggi.

Passaggio 1: aggiungere la dipendenza richiesta

Configura le dipendenze del progetto (ad esempio, nel tuo `build.gradle` file `pom.xml` o) per utilizzare la versione `2.30.3` o successiva di. AWS SDK per Java

Includi l'`emf-metric-logging-publisher`ArtifactID con il `2.30.3` numero di versione o successivo nelle dipendenze del progetto.

Per esempio:

```
<project>
  <dependencyManagement>
   <dependencies>
      <dependency>
        <groupId>software.amazon.awssdk</groupId>
        <artifactId>bom</artifactId>
        <version>2.30.11</version>  <!-- Navigate the link to see the latest version.
 -->
        <type>pom</type>
        <scope>import</scope>
      </dependency>
   </dependencies>
  </dependencyManagement>
  <dependencies>
   <dependency>
      <groupId>software.amazon.awssdk</groupId>
      <artifactId>emf-metric-logging-publisher</artifactId>
   </dependency>
  </dependencies>
</project>
```

Passaggio 2: configura le autorizzazioni richieste

Abilita `logs:PutLogEvents` le autorizzazioni per l'identità IAM utilizzata dall'editore delle metriche per consentire all'SDK for Java di scrivere log in formato EMF.

Fase 3: Configurazione della registrazione

Per garantire una corretta raccolta delle metriche, configura la registrazione in modo che venga inviata alla console a un `INFO` livello o inferiore (ad esempio). `DEBUG` Nel tuo `log4j2.xml` file:

```
<Loggers>
  <Root level="WARN">
   <AppenderRef ref="ConsoleAppender"/>
  </Root>
  <Logger
 name="software.amazon.awssdk.metrics.publishers.emf.EmfMetricLoggingPublisher"
 level="INFO" />
</Loggers>
```

Per ulteriori informazioni su come configurare un `log4j2.xml` file, consulta l'argomento relativo alla registrazione in questa guida.

Configurazione e utilizzo **`EmfMetricLoggingPublisher`**

La seguente classe di funzioni Lambda crea e configura prima un'`EmfMetricLoggingPublisher`istanza e poi la utilizza con un client di servizio Amazon DynamoDB:

```
public class GameIdHandler implements RequestHandler<Map<String, String>, String> {
    private final EmfMetricLoggingPublisher emfPublisher;
    private final DynamoDbClient dynamoDb;

    public GameIdHandler() {
        // Build the publisher.
        this.emfPublisher = EmfMetricLoggingPublisher.builder()
                .namespace("namespace")
                .dimensions(CoreMetric.SERVICE_ID,
                        CoreMetric.OPERATION_NAME)
                .build();
        // Add the publisher to the client.
        this.dynamoDb = DynamoDbClient.builder()
                .overrideConfiguration(c -> c.addMetricPublisher(emfPublisher))
                .region(Region.of(System.getenv("AWS_REGION")))
                .build();
    }

    @Override
    public String handleRequest(Map<String, String> event, Context context) {
        Map<String, AttributeValue> gameItem = new HashMap<>();

        gameItem.put("gameId", AttributeValue.builder().s(event.get("id")).build());

        PutItemRequest putItemRequest = PutItemRequest.builder()
                .tableName("games")
                .item(gameItem)
                .build();

        dynamoDb.putItem(putItemRequest);

        return "Request handled";
    }
}
```

Quando il client DynamoDB esegue il metodo, pubblica automaticamente `putItem` le metriche in un flusso di log in formato EMF. CloudWatch

La documentazione dell'API `EmfMetricLoggingPublisher.Builder` mostra le opzioni di configurazione che è possibile utilizzare.

È inoltre possibile abilitare la registrazione delle metriche EMF per una singola richiesta, come mostrato per. CloudWatchMetricPublisher

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 l'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	Tipo
ApiCallDuration	Il tempo totale impiegato per completare una richiesta (inclusi tutti i nuovi tentativi).	Durata*
ApiCallSuccessful	Vero se la chiamata API ha avuto successo; falso in caso contrario.	Booleano
CredentialsFetchDuration	Il tempo impiegato per recuperare le credenziali di AWS firma per la richiesta.	Durata*
EndpointResolveDuration	Il tempo impiegato per risolvere l'endpoint utilizzato per la chiamata API.	Durata*
MarshallingDuration	Il tempo necessario per convertire una richiesta SDK in una richiesta HTTP.	Durata*
OperationName	Il nome dell' AWS API a cui viene effettuata la richiesta.	Stringa

Nome parametro	Descrizione	Tipo
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*

* java.time.Duration.

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	Tipo
AwsExtendedRequestId	L'ID della richiesta estesa della richiesta di servizio.	Stringa
AwsRequestId	L'ID della richiesta di servizio.	Stringa
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 in byte/secondo.	Doppio

Nome parametro	Descrizione	Tipo
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*

* java.time.Duration.

Metriche HTTP

Nome parametro	Descrizione	Tipo	Client HTTP richiesto*
AvailableConcurrency	Il numero di richieste concorrenti rimanenti che possono essere	Numero intero	Apache, Netty, CRT

Nome parametro	Descrizione	Tipo	Client HTTP richiesto*
	supportate dal client HTTP senza dover stabilire un'altra connessione.		
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 supportate dal client HTTP.	Numero intero	Apache, Netty, CRT

Nome parametro	Descrizione	Tipo	Client HTTP richiesto*
PendingConcurrencyAcquires	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

* java.time.Duration.

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 URLConnection basato () `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

- Lavora con CloudWatch
- AWS servizi di database e AWS SDK for Java 2.x
- Lavora con DynamoDB
- Lavora con Amazon EC2
- Lavora con IAM
- Lavora con Kinesis
- Funzioni di richiamo, elenco ed eliminazione AWS Lambda
- Funziona con Amazon S3
- Lavora con Amazon Simple Notification Service
- Lavora con Amazon Simple Queue Service
- Lavora con Amazon Transcribe

Lavora con CloudWatch

Questa sezione fornisce esempi di programmazione di Amazon CloudWatch utilizzando AWS SDK per Java 2.x.

Amazon CloudWatch monitora le tue risorse Amazon Web Services (AWS) e le applicazioni su cui esegui AWS in tempo reale. Puoi utilizzarlo CloudWatch per raccogliere e tenere traccia delle metriche, che sono variabili che puoi misurare per le tue risorse e applicazioni. CloudWatch gli allarmi 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 CloudWatch utente.

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 ListMetricsResponsechiamando 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

- ListMetricsnell' 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 CloudWatch utente.
- AWS Namespace nella Guida per l'utente. Amazon CloudWatch
- PutMetricDatanell'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 PutMetricAlarmRequestcompilazione 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 DescribeAlarmsRequestche 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 desCWAlarms( CloudWatchClient cw) {

        try {

            boolean done = false;
            String newToken = null;

            while(!done) {
                DescribeAlarmsResponse response;

                if (newToken == null) {
```

```
                    DescribeAlarmsRequest request =
DescribeAlarmsRequest.builder().build();
                    response = cw.describeAlarms(request);
                } else {
                    DescribeAlarmsRequest request = DescribeAlarmsRequest.builder()
                        .nextToken(newToken)
                        .build();
                    response = cw.describeAlarms(request);
                }

                for(MetricAlarm alarm : response.metricAlarms()) {
                    System.out.printf("\n Retrieved alarm %s", alarm.alarmName());
                }

                if(response.nextToken() == null) {
                    done = true;
                } else {
                    newToken = response.nextToken();
                }
            }

        } catch (CloudWatchException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        System.out.printf("Done");
    }
```

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's. 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 DeleteAlarmsRequestcontenente 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

- Uso degli Amazon CloudWatch allarmi nella Guida per l' Amazon CloudWatch utente
- PutMetricAlarmnell' Amazon CloudWatch API Reference
- DescribeAlarmsnell' Amazon CloudWatch API Reference
- DeleteAlarmsnell' Amazon CloudWatch API 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
- PutEventsnell'Amazon EventBridge API Reference
- PutTargetsnell'Amazon EventBridge API Reference
- PutRulenell'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 un APIs servizio Web per AWS la gestione della risorsa (database) e di un APIs servizio Web per interagire con i dati. Nell'SDK for Java 2.x questi tipi di servizi dispongono di client di servizio dedicati, ad esempio Dynamo. DBClient

Altri servizi di database dispongono di servizi Web APIs 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 DocDbClientcorrispondente 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	Lavorare con il database
Client di servizio SDK: DynamoDbClient	Client di servizio SDK: DynamoDbClient
Esempio: applicazione REST React/Spring che utilizza DynamoDB	CreateTableEsempi ListTables:, DeleteTable
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»)	

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: RdsClient
Esempio: applicazione REST React/Spring che utilizza MySQL	Esempi RdsClient : diversi esempi

Esempi di Amazon Redshift

Lavorare con i dati	Lavorare con il database
Client di servizio SDK: RedshiftDataClient	Client di servizio SDK: RedshiftClient
Esempi: diversi RedshiftDataClient esempi	Esempi: diversi RedshiftClient esempi
Esempio: applicazione REST React/Spring che utilizza RedshiftDataClient	

Esempi di Amazon Aurora Serverless v2

Lavorare con i dati	Lavorare con il database
Client di servizio SDK: RdsDataClient	Client di servizio SDK: RdsClient
Esempio: applicazione REST React/Spring che utilizza RdsDataClient	Esempi: diversi esempi RdsClient

Esempi di Amazon DocumentDB

Lavorare con i dati	Lavorare con il database
API non SDK: libreria Java specifica per MongoDB (ad esempio MongoDB per Java); il codice gestisce le connessioni al database o un pool di connessioni.	Client di servizio SDK: DocDbClient
Esempi: DocumentDB (Mongo) Developer Guide (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 per 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”

Utilizza endpoint basati sull' AWS account

DynamoDB AWS offre endpoint basati su account che possono migliorare le prestazioni utilizzando l'ID dell'account per semplificare AWS il routing delle richieste.

Per sfruttare questa funzionalità, è necessario utilizzare la versione 2.28.4 o successiva della versione 2 di. AWS SDK per Java Puoi trovare l'ultima versione dell'SDK elencata nell'archivio centrale di Maven. Dopo che una versione supportata di SDK è attiva, utilizza automaticamente i nuovi endpoint.

Se desideri disattivare il routing basato sull'account, hai quattro opzioni:

- Configurare un client di servizio DynamoDB con `AccountIdEndpointMode` l'impostazione su. `DISABLED`
- Imposta una variabile di ambiente.
- Imposta una proprietà del sistema JVM.
- Aggiorna l'impostazione del file di AWS configurazione condiviso.

Il seguente frammento è un esempio di come disabilitare il routing basato su account configurando un client di servizio DynamoDB:

```
DynamoDbClient.builder()
              .accountIdEndpointMode(AccountIdEndpointMode.DISABLED)
              .build();
```

La AWS SDKs and Tools Reference Guide fornisce ulteriori informazioni sulle ultime tre opzioni di configurazione.

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 composita, 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, `DynamoDbException` viene generata a.

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.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 "";
    }
```

Vedi l'esempio completo su. GitHub

Creazione di una tabella con una chiave primaria composita

L'esempio seguente crea una tabella con due attributi. Entrambi gli attributi vengono utilizzati per la chiave primaria composita.

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 describeDymamoDBTable(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, ResourceNotFoundExceptionviene 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, ResourceNotFoundExceptionviene 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, un elemento è una raccolta di attributi, ognuno dei quali ha 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 GetItemRequestoggetto con il nome della tabella e il valore della chiave primaria dell'elemento desiderato. Restituisce un GetItemResponseoggetto 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 GetItemRequestoggetto 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, ResourceNotFoundExceptionviene 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, ResourceNotFoundExceptionviene 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, ResourceNotFoundExceptionviene 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 contenuti DynamoDB nella 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")
    ...
}
```

Esegui una ricerca nell'archivio centrale di Maven per la versione più recente e sostituiscila con questo valore. *<VERSION>*

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 static `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 DynamoDbEnhancedClientclasse, 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 a. `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 esiste `Customer` 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 lambda`builder`, 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.enhanced.dynamodb.model.CreateTableEnhancedRequest;
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 il `putItem` metodo, il client avanzato non include gli attributi con valori nulli di un oggetto dati mappato nella richiesta a DynamoDB.

Il comportamento predefinito dell'SDK per `updateItem` le richieste rimuove gli attributi dall'elemento in DynamoDB che sono impostati su null nell'oggetto inviato nel metodo. `updateItem` Se intendi aggiornare alcuni valori degli attributi e mantenere invariati gli altri, hai due opzioni.

- Recuperate l'elemento (utilizzando`getItem`) prima di apportare modifiche ai valori. Utilizzando questo approccio, l'SDK invia tutti i valori vecchi e aggiornati a DynamoDB.
- Usa `IgnoreNullsMode.SCALAR_ONLY` o `IgnoreNullsMode.MAPS_ONLY` quando crei la richiesta per aggiornare l'elemento. Entrambe le modalità ignorano le proprietà con valori nulli nell'oggetto che rappresentano gli attributi scalari in DynamoDB. L'the section called “Aggiorna gli elementi che contengono tipi complessi”argomento di questa guida contiene ulteriori informazioni sui `IgnoreNullsMode` valori e su come lavorare con tipi complessi.

L'esempio seguente illustra `ignoreNullsMode()` il `updateItem()` metodo.

```
    public static void updateItemNullsExample() {
        Customer customer = new Customer();
        customer.setCustName("CustomerName");
        customer.setEmail("email");
        customer.setId("1");
        customer.setRegistrationDate(Instant.now());

        logger.info("Original customer: {}", customer);

        // Put item with values for all attributes.
        try {
            customerAsyncDynamoDbTable.putItem(customer).join();
        } catch (RuntimeException rte) {
```

```
            logger.error("A exception occurred during putItem: {}",
rte.getCause().getMessage(), rte);
            return;
        }

        // Create a Customer instance with the same 'id' and 'email' values, but a
different 'name' value.
        // Do not set the 'registrationDate' attribute.
        Customer customerForUpdate = new Customer();
        customerForUpdate.setCustName("NewName");
        customerForUpdate.setEmail("email");
        customerForUpdate.setId("1");

        // Update item without setting the 'registrationDate' property and set
IgnoreNullsMode to SCALAR_ONLY.
        try {
            Customer updatedWithNullsIgnored = customerAsyncDynamoDbTable.updateItem(b
-> b
                            .item(customerForUpdate)
                            .ignoreNullsMode(IgnoreNullsMode.SCALAR_ONLY))
                    .join();
            logger.info("Customer updated with nulls ignored: {}",
updatedWithNullsIgnored.toString());
        } catch (RuntimeException rte) {
            logger.error("An exception occurred during updateItem: {}",
rte.getCause().getMessage(), rte);
            return;
        }

        // Update item without setting the registrationDate attribute and not setting
ignoreNulls to true.
        try {
            Customer updatedWithNullsUsed =
customerAsyncDynamoDbTable.updateItem(customerForUpdate)
                    .join();
            logger.info("Customer updated with nulls used: {}",
updatedWithNullsUsed.toString());
        } catch (RuntimeException rte) {
            logger.error("An exception occurred during updateItem: {}",
rte.getCause().getMessage(), rte);
        }
    }
```

```
// Logged lines.
Original customer: Customer [id=1, name=CustomerName, email=email,
 regDate=2024-10-11T14:12:30.222858Z]
Customer updated with nulls ignored: Customer [id=1, name=NewName, email=email,
 regDate=2024-10-11T14:12:30.222858Z]
Customer updated with nulls used: Customer [id=1, name=NewName, email=email,
 regDate=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

Entrambi i APIs client DynamoDB, standard e avanzato, consentono di lavorare con le tabelle DynamoDB per eseguire operazioni CRUD (creazione, lettura, aggiornamento ed eliminazione) a livello di dati. La differenza tra i client sta nel modo in cui ciò viene realizzato. APIs Utilizzando il client standard, si lavora 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 entrambi i client APIs 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 dai due client APIs, 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 immutable`Customer`.
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 createdDate;
}
```

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()
```

`Expression`Di 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 AttributeValuesclasse dispone di metodi pratici per generare un'`AttributeValue`istanza da un valore letterale. In alternativa, è possibile utilizzare il `AttributeValue.Builder` per generare un'`AttributeValue`istanza.

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'`putItem`operazione per verificare se nel database esiste già un elemento con un valore dell'`movie`attributo uguale all'attributo dell'`movie`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 `deleteItem()` metodi`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 “`Query`esempi 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, UpdateExpressionsfornisci 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 PageIterableoggetto, 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 la scansione e l'interrogazione. APIs

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. ProductCatalogÈ l'oggetto modello che è stato mostrato in precedenza.

L'espressione di filtro mostrata dopo la riga di commento 2 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 1.

Dopo la riga di commento 3, l'`PageIterable`oggetto,`pagedResults`, 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 4a scorre attraverso ogni pagina e ordina e registra gli elementi su ogni pagina. La versione successiva alla riga di commento 4b 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.Iterable`SdkIterable `Iterable`porta i `spliterator` metodi`forEach`, `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'SdkPublisherinterfaccia. 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 per prezzo e registra ogni articolo.

La `ProductCatalogSubscriber` classe CountDownLatchin 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 l'`scan`API, l'`query`API 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 composita 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 composita 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 QueryConditionalistanze. `QueryConditionals`funzionano 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`**riga di commento 2.

Dopo la riga di commento 3, l'esempio mostra l'QueryEnhancedRequestistanza 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 composita, tutte le `QueryConditional` istanze richiedono un valore di chiave di partizione. `QueryConditional`i 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 `DynamoDbEnhancedClient.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`dati `Customer`e 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` builder che accetta un valore di chiave primaria e un valore di chiave di ordinamento per specificare l'elemento da leggere. Se la classe di dati ha una chiave composita, è necessario fornire sia il valore della chiave di partizione che il valore della chiave di ordinamento.

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 value and sort key value 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 `ProductCatalog`e `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 `ProductCatalog`e `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 non `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` è1982, 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 SubjectLastPostedDateIndexutilizza l'`Subject`attributo per la chiave di partizione e il `LastPostedDateTime` per la chiave di ordinamento. L'LSI denominato ForumLastPostedDateIndexutilizza 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` Il 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 QueryConditionaloggetto 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 vengono 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 a `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. `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

`TableSchema`I 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 un`StaticTableSchema`, 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 di`TableSchema`, non si definiscono gli attributi per un'`DocumentTableSchema`istanza. Di solito, si specificano solo chiavi primarie e fornitori di convertitori di attributi. Un'`EnhancedDocument`istanza 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) { this.internalKey = internalKey;}
```

Includi gli attributi

Per modificare il nome di un attributo utilizzato nella tabella DynamoDB, contrassegnalo 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) { 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'AttributeConverterinterfaccia 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 an `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`gli 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. `HttpCookeConverter`

```
   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'`HttpCookie`istanza 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, desideri 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 appiattire la gerarchia.

Usa fagioli annotati

Per l'approccio all'annotazione, entrambe le classi devono contenere l'`@DynamoDbBean`annotazione 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 createdDate;

    @DynamoDbPartitionKey
    public String getId() { return id; }
    public void setId(String id) { this.id = id; }

    public String getCreatedDate() { return createdDate; }
    public void setCreatedDate(String createdDate) { this.createdDate =
 createdDate; }
}
```

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 createdDate;
}
```

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 createdDate;

    public String getId() { return id; }
    public void setId(String id) { this.id = id; }

    public String getCreatedDate() { return createdDate; }
```

```
    public void setCreatedDate(String createdDate) { this.createdDate =
 createdDate; }
```

Lombok

```
@Data
@ToString(callSuper = true)
public class Customer extends GenericRecord{
    private String name;
}

@Data
public abstract class GenericRecord {
    private String id;
    private String createdDate;
}
```

Usa la composizione

Se le tue classi usano la composizione, usa i seguenti approcci per appiattire 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 createdDate;

    @DynamoDbPartitionKey
    public String getId() { return this.id; }
    public void setId(String id) { this.id = id; }

    public String getCreatedDate() { return this.createdDate; }
    public void setCreatedDate(String createdDate) { this.createdDate =
 createdDate; }
}
```

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 createdDate;
}
```

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::getCreatedDate)
                                .setter(GenericRecord::setCreatedDate))
                        .build();

        StaticTableSchema<Customer> CUSTOMER_SCHEMA =
                StaticTableSchema.builder(Customer.class)
                        .newItemSupplier(Customer::new)
                        .addAttribute(String.class, a -> a.name("name")
                                .getter(Customer::getName)
                                .setter(Customer::setName))
                        // Because we are flattening a component object, we supply a
getter and setter so the
                        // mapper knows how to access it.
                        .flatten(GENERIC_RECORD_SCHEMA, Customer::getRecord,
Customer::setRecord)
                        .build();
```

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 createdDate;

    public String getId() { return this.id; }
    public void setId(String id) { this.id = id; }

    public String getCreatedDate() { return this.createdDate; }
    public void setCreatedDate(String createdDate) { this.createdDate =
 createdDate; }
}
```

Lombok

```
@Data
public class Customer {
    private String name;
    private GenericRecord record;
}

@Data
public class GenericRecord {
    private String id;
    private String createdDate;
}
```

Puoi utilizzare il modello builder per appiattire 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
`{ "id": "1", "createdDate": "today", "name": "my name" }`	`{ "id": "1", "record": { "createdDate": "today", "name": "my name" } }`

La differenza diventa importante se si dispone di altro codice che accede alla tabella DynamoDB che prevede di trovare determinati attributi.

Lavora con attributi che sono bean, mappe, elenchi e set

Una definizione di bean, come la `Person` classe mostrata di seguito, potrebbe definire proprietà (o attributi) che si riferiscono a tipi con attributi aggiuntivi. Ad esempio, nella `Person` classe, `mainAddress` c'è una proprietà che si riferisce a un `Address` bean che definisce attributi di valore aggiuntivi. `addresses`si riferisce a una mappa Java, i cui elementi si riferiscono ai `Address` bean. Questi tipi complessi possono essere considerati contenitori di attributi semplici da utilizzare per il valore dei dati nel contesto di DynamoDB.

DynamoDB si riferisce alle proprietà di valore degli elementi nidificati, come mappe, elenchi o bean, come attributi nidificati. La Amazon DynamoDB Developer Guide si riferisce al formato salvato di una mappa, un elenco o un bean Java come tipo di documento. Gli attributi semplici utilizzati per il loro valore di dati in Java sono denominati tipi scalari in DynamoDB. I set, che contengono più elementi scalari dello stesso tipo e sono denominati tipi di set.

È importante sapere che l'API DynamoDB Enhanced Client converte una proprietà che è bean in un tipo di documento di mappa DynamoDB quando viene salvata.

Classe **`Person`**

```
@DynamoDbBean
public class Person {
    private Integer id;
    private String firstName;
    private String lastName;
    private Integer age;
    private Address mainAddress;
    private Map<String, Address> addresses;
    private List<PhoneNumber> phoneNumbers;
    private Set<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 Address getMainAddress() {
        return mainAddress;
    }

    public void setMainAddress(Address mainAddress) {
        this.mainAddress = mainAddress;
    }

    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 Set<String> getHobbies() {
        return hobbies;
    }

    public void setHobbies(Set<String> hobbies) {
        this.hobbies = hobbies;
    }

    @Override
    public String toString() {
        return "Person{" +
               "addresses=" + addresses +
               ", id=" + id +
               ", firstName='" + firstName + '\'' +
               ", lastName='" + lastName + '\'' +
               ", age=" + age +
               ", mainAddress=" + mainAddress +
               ", 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 + '\'' +
                '}';
    }
}
```

Salva tipi complessi

Usa classi di dati annotate

Puoi salvare gli attributi annidati per le classi personalizzate semplicemente annotandoli. 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 astratti con i builder

L'approccio alternativo consiste nell'utilizzare generatori di schemi di tabelle statiche per ogni classe bean annidata, 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.documentOf(Address.class, TABLE_SCHEMA_ADDRESS),
a -> a.name("mainAddress")
               .getter(Person::getMainAddress)
               .setter(Person::setMainAddress))
           .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 di progetto di tipi complessi

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> hobbiesScanResult = personDynamoDbTable.scan(b -> b
                // Use the 'attributesToProject()' method to access first-level
 attributes.
                .attributesToProject("hobbies"));

        hobbiesScanResult.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]}
```

Usa tipi complessi nelle espressioni

È possibile utilizzare tipi complessi nelle espressioni, ad esempio espressioni di filtro ed espressioni di condizione, utilizzando gli operatori di dereferenziamento per navigare nella struttura del tipo complesso. Per oggetti e mappe, usa `.` (`dot`) e per gli elementi dell'elenco `[n]` (parentesi quadre attorno al numero di sequenza dell'elemento). Non puoi fare riferimento ai singoli elementi di un set, ma puoi usare la `contains`funzione.

L'esempio seguente mostra due espressioni di filtro utilizzate nelle operazioni di scansione. Le espressioni di filtro specificano le condizioni di corrispondenza per gli elementi da inserire nei risultati. L'esempio utilizza e `Person` `Address` le `PhoneNumber` classi mostrate in precedenza.

```
   public void scanUsingFilterOfNestedAttr() {
       // The following is a filter expression for an attribute that is a map of
Address objects.
       // By using this filter expression, the SDK returns Person objects that have an
address
       // with 'mailing' as a key and 'MS2' for a state value.
       Expression addressFilter = Expression.builder()
               .expression("addresses.#type.#field = :value")
```

```
                .putExpressionName("#type", "mailing")
                .putExpressionName("#field", "state")
                .putExpressionValue(":value",
AttributeValue.builder().s("MS2").build())
                .build();

        PageIterable<Person> addressFilterResults = personDynamoDbTable.scan(rb -> rb.
                filterExpression(addressFilter));
        addressFilterResults.items().stream().forEach(p -> logger.info("Person: {}",
p));

        assert addressFilterResults.items().stream().count() == 1;


        // The following is a filter expression for an attribute that is a list of
phone numbers.
        // By using this filter expression, the SDK returns Person objects whose second
phone number
        // in the list has a type equal to 'cell'.
        Expression phoneFilter = Expression.builder()
                .expression("phoneNumbers[1].#type = :type")
                .putExpressionName("#type", "type")
                .putExpressionValue(":type",
AttributeValue.builder().s("cell").build())
                .build();

        PageIterable<Person> phoneFilterResults = personDynamoDbTable.scan(rb -> rb
                .filterExpression(phoneFilter)
                .attributesToProject("id", "firstName", "lastName", "phoneNumbers")
        );

        phoneFilterResults.items().stream().forEach(p -> logger.info("Person: {}", p));

        assert phoneFilterResults.items().stream().count() == 1;
        assert
phoneFilterResults.items().stream().findFirst().get().getPhoneNumbers().get(1).getType().equal
    }
```

Metodo di supporto che popola la tabella

```
    public static void populateDatabase() {
        Person person1 = new Person();
        person1.setId(1);
```

```
        person1.setFirstName("FirstName1");
        person1.setLastName("LastName1");

        Address billingAddr1 = new Address();
        billingAddr1.setState("BS1");
        billingAddr1.setCity("BillingTown1");

        Address mailing1 = new Address();
        mailing1.setState("MS1");
        mailing1.setCity("MailingTown1");

        person1.setAddresses(Map.of("billing", billingAddr1, "mailing", mailing1));

        PhoneNumber pn1_1 = new PhoneNumber();
        pn1_1.setType("work");
        pn1_1.setNumber("111-111-1111");

        PhoneNumber pn1_2 = new PhoneNumber();
        pn1_2.setType("home");
        pn1_2.setNumber("222-222-2222");

        List<PhoneNumber> phoneNumbers1 = List.of(pn1_1, pn1_2);
        person1.setPhoneNumbers(phoneNumbers1);

        personDynamoDbTable.putItem(person1);

        Person person2 = person1;
        person2.setId(2);
        person2.setFirstName("FirstName2");
        person2.setLastName("LastName2");

        Address billingAddress2 = billingAddr1;
        billingAddress2.setCity("BillingTown2");
        billingAddress2.setState("BS2");

        Address mailing2 = mailing1;
        mailing2.setCity("MailingTown2");
        mailing2.setState("MS2");

        person2.setAddresses(Map.of("billing", billingAddress2, "mailing", mailing2));

        PhoneNumber pn2_1 = new PhoneNumber();
        pn2_1.setType("work");
        pn2_1.setNumber("333-333-3333");
```

```
        PhoneNumber pn2_2 = new PhoneNumber();
        pn2_2.setType("cell");
        pn2_2.setNumber("444-444-4444");

        List<PhoneNumber> phoneNumbers2 = List.of(pn2_1, pn2_2);
        person2.setPhoneNumbers(phoneNumbers2);

        personDynamoDbTable.putItem(person2);
    }
```

Rappresentazione JSON degli elementi del database

```
{
 "id": 1,
 "addresses": {
  "billing": {
   "city": "BillingTown1",
   "state": "BS1",
   "street": null,
   "zipCode": null
  },
  "mailing": {
   "city": "MailingTown1",
   "state": "MS1",
   "street": null,
   "zipCode": null
  }
 },
 "firstName": "FirstName1",
 "lastName": "LastName1",
 "phoneNumbers": [
  {
   "number": "111-111-1111",
   "type": "work"
  },
  {
   "number": "222-222-2222",
   "type": "home"
  }
 ]
}
```

```
{
 "id": 2,
 "addresses": {
  "billing": {
   "city": "BillingTown2",
   "state": "BS2",
   "street": null,
   "zipCode": null
  },
  "mailing": {
   "city": "MailingTown2",
   "state": "MS2",
   "street": null,
   "zipCode": null
  }
 },
 "firstName": "FirstName2",
 "lastName": "LastName2",
 "phoneNumbers": [
  {
   "number": "333-333-3333",
   "type": "work"
  },
  {
   "number": "444-444-4444",
   "type": "cell"
  }
 ]
}
```

Aggiorna gli elementi che contengono tipi complessi

Per aggiornare un elemento che contiene tipi complessi, sono disponibili due approcci di base:

- Approccio 1: recupera prima l'elemento (utilizzando`getItem`), aggiorna l'oggetto, quindi chiama`DynamoDbTable#updateItem`.
- Approccio 2: non recuperate l'elemento, ma create una nuova istanza, impostate le proprietà che desiderate aggiornare e inviate l'istanza `DynamoDbTable#updateItem` impostando il valore appropriato di. `IgnoreNullsMode` Questo approccio non richiede il recupero dell'elemento prima di aggiornarlo.

Gli esempi mostrati in questa sezione utilizzano le `PhoneNumber` classi `PersonAddress`, e mostrate in precedenza.

Approccio di aggiornamento 1: recupera, quindi aggiorna

Utilizzando questo approccio, vi assicurate che nessun dato vada perso durante l'aggiornamento. L'API DynamoDB Enhanced Client ricrea il bean con gli attributi dell'elemento salvato in DynamoDB, inclusi i valori di tipi complessi. È quindi necessario utilizzare i getter e i setter per aggiornare il bean. Lo svantaggio di questo approccio è il costo che si deve sostenere per recuperare prima l'articolo.

L'esempio seguente dimostra che non si perde alcun dato se si recupera l'elemento prima di aggiornarlo.

```
    public void retrieveThenUpdateExample()  {
        // Assume that we ran this code yesterday.
        Person person = new Person();
        person.setId(1);
        person.setFirstName("FirstName");
        person.setLastName("LastName");

        Address mainAddress = new Address();
        mainAddress.setStreet("123 MyStreet");
        mainAddress.setCity("MyCity");
        mainAddress.setState("MyState");
        mainAddress.setZipCode("MyZipCode");
        person.setMainAddress(mainAddress);

        PhoneNumber homePhone = new PhoneNumber();
        homePhone.setNumber("1111111");
        homePhone.setType("HOME");
        person.setPhoneNumbers(List.of(homePhone));

        personDynamoDbTable.putItem(person);

        // Assume that we are running this code now.
        // First, retrieve the item
        Person retrievedPerson =
personDynamoDbTable.getItem(Key.builder().partitionValue(1).build());

        // Make any updates.
        retrievedPerson.getMainAddress().setCity("YourCity");
```

```
        // Save the updated bean. 'updateItem' returns the bean as it appears after the
 update.
        Person updatedPerson = personDynamoDbTable.updateItem(retrievedPerson);

        // Verify for this example.
        Address updatedMainAddress = updatedPerson.getMainAddress();
        assert updatedMainAddress.getCity().equals("YourCity");
        assert updatedMainAddress.getState().equals("MyState"); // Unchanged.
        // The list of phone numbers remains; it was not set to null;
        assert updatedPerson.getPhoneNumbers().size() == 1;
    }
```

Approccio di aggiornamento 2: utilizzare un **`IgnoreNullsMode`** enum senza prima recuperare l'elemento

Per aggiornare un elemento in DynamoDB, puoi fornire un nuovo oggetto con solo le proprietà che desideri aggiornare e lasciare gli altri valori come nulli. Con questo approccio, è necessario essere consapevoli di come i valori nulli nell'oggetto vengono trattati dall'SDK e di come è possibile controllarne il comportamento.

Per specificare quali proprietà con valori nulli vuoi che l'SDK ignori, fornisci un enum quando crei il. `IgnoreNullsMode` `UpdateItemEnhancedRequest` Come esempio di utilizzo di uno dei valori enumerati, il frammento seguente utilizza la modalità. `IgnoreNullsMode.SCALAR_ONLY`

```
// Create a new Person object to update the existing item in DynamoDB.
Person personForUpdate = new Person();
personForUpdate.setId(1);
personForUpdate.setFirstName("updatedFirstName");  // 'firstName' is a top scalar
 property.

Address addressForUpdate = new Address();
addressForUpdate.setCity("updatedCity");
personForUpdate.setMainAddress(addressForUpdate);

personDynamoDbTable.updateItem(r -> r
                .item(personForUpdate)
                .ignoreNullsMode(IgnoreNullsMode.SCALAR_ONLY));

/* With IgnoreNullsMode.SCALAR_ONLY provided, The SDK ignores all null properties. The
 SDK adds or replaces
the 'firstName' property with the provided value, "updatedFirstName". The SDK updates
 the 'city' value of
```

```
'mainAddress', as long as the 'mainAddress' attribute already exists in DynamoDB.

In the background, the SDK generates an update expression that it sends in the request
 to DynamoDB.
The following JSON object is a simplified version of what it sends. Notice that the SDK
 includes the paths
to 'mainAddress.city' and 'firstName' in the SET clause of the update expression. No
 null values in
'personForUpdate' are included.

{
  "TableName": "PersonTable",
  "Key": {
    "id": {
      "N": "1"
    }
  },
  "ReturnValues": "ALL_NEW",
  "UpdateExpression": "SET #mainAddress.#city = :mainAddress_city, #firstName
 = :firstName",
  "ExpressionAttributeNames": {
    "#city": "city",
    "#firstName": "firstName",
    "#mainAddress": "mainAddress"
  },
  "ExpressionAttributeValues": {
    ":firstName": {
      "S": "updatedFirstName"
    },
    ":mainAddress_city": {
      "S": "updatedCity"
    }
  }
}

Had we chosen 'IgnoreNullsMode.DEFAULT' instead of 'IgnoreNullsMode.SCALAR_ONLY', the
 SDK would have included
null values in the "ExpressionAttributeValues" section of the request as shown in the
 following snippet.

  "ExpressionAttributeValues": {
    ":mainAddress": {
      "M": {
        "zipCode": {
```

```
          "NULL": true
        },
        "city": {
          "S": "updatedCity"
        },
        "street": {
          "NULL": true
        },
        "state": {
          "NULL": true
        }
      }
    },
    ":firstName": {
      "S": "updatedFirstName"
    }
  }
*/
```

L'Amazon DynamoDB Developer Guide contiene ulteriori informazioni sulle espressioni di aggiornamento.

Descrizioni delle opzioni **`IgnoreNullsMode`**

- `IgnoreNullsMode.SCALAR_ONLY`- Utilizzate questa impostazione per aggiornare gli attributi scalari a qualsiasi livello. L'SDK crea un'istruzione di aggiornamento che invia solo attributi scalari non nulli a DynamoDB. L'SDK ignora gli attributi scalari con valori nulli di un bean o di una mappa, mantenendo il valore salvato in DynamoDB.

  Quando si aggiorna un attributo scalare di map o bean, la mappa deve già esistere in DynamoDB. Se aggiungi una mappa o un bean all'oggetto che non esiste già per l'oggetto in DynamoDB, ricevi `DynamoDbException` un messaggio con il messaggio Il percorso del documento fornito nell'espressione di aggiornamento non è valido per l'aggiornamento. È necessario utilizzare `MAPS_ONLY` la modalità per aggiungere un bean o una map a DynamoDB prima di aggiornare uno qualsiasi dei suoi attributi.
- `IgnoreNullsMode.MAPS_ONLY`- Utilizzate questa impostazione per aggiungere o sostituire proprietà che sono un bean o una mappa. L'SDK sostituisce o aggiunge qualsiasi mappa o bean fornito nell'oggetto. Tutti i bean o le mappe che sono nulli nell'oggetto vengono ignorati, mantenendo la mappa esistente in DynamoDB.

- `IgnoreNullsMode.DEFAULT`- Con questa impostazione, l'SDK non ignora mai i valori nulli. Gli attributi scalari a qualsiasi livello che sono nulli vengono aggiornati a null. L'SDK aggiorna qualsiasi proprietà bean, map, list o set con valori nulli nell'oggetto su null in DynamoDB. Quando si utilizza questa modalità, o non si fornisce una modalità poiché è la modalità predefinita, è necessario recuperare prima l'elemento in modo che i valori in DynamoDB non siano impostati su null forniti nell'oggetto per l'aggiornamento, a meno che non si intenda impostare i valori su null.

In tutte le modalità, se si fornisce un oggetto con un elenco o un set non nullo, l'elenco o il set viene salvato in DynamoDB. `updateItem`

Perché le modalità?

Quando fornisci un oggetto con un bean o una map al `updateItem` metodo, l'SDK non è in grado di stabilire se utilizzare i valori delle proprietà nel bean (o i valori di immissione nella mappa) per aggiornare l'elemento o se l'intero bean/map debba sostituire ciò che è stato salvato in DynamoDB.

Partendo dal nostro esempio precedente che mostra innanzitutto il recupero dell'elemento, proviamo ad aggiornare l'attributo di without the retrieval. `city` `mainAddress`

```
/* The retrieval example saved the Person object with a 'mainAddress' property whose
 'city' property value is "MyCity".
/* Note that we create a new Person with only the necessary information to update the
 city value
of the mainAddress. */
Person personForUpdate = new Person();
personForUpdate.setId(1);
// The update we want to make changes the city.
Address mainAddressForUpdate = new Address();
mainAddressForUpdate.setCity("YourCity");
personForUpdate.setMainAddress(mainAddressForUpdate);

// Lets' try the following:
Person updatedPerson = personDynamoDbTable.updateItem(personForUpdate);
/*
 Since we haven't retrieved the item, we don't know if the 'mainAddress' property
 already exists, so what update expression should the SDK generate?

A) Should it replace or add the 'mainAddress' with the provided object (setting all
 attributes to null other than city)
   as shown in the following simplified JSON?
```

```
      {
        "TableName": "PersonTable",
        "Key": {
          "id": {
            "N": "1"
          }
        },
        "ReturnValues": "ALL_NEW",
        "UpdateExpression": "SET #mainAddress = :mainAddress",
        "ExpressionAttributeNames": {
          "#mainAddress": "mainAddress"
        },
        "ExpressionAttributeValues": {
          ":mainAddress": {
            "M": {
              "zipCode": {
                "NULL": true
              },
              "city": {
                "S": "YourCity"
              },
              "street": {
                "NULL": true
              },
              "state": {
                "NULL": true
              }
            }
          }
        }
      }

B) Or should it update only the 'city' attribute of an existing 'mainAddress' as shown
 in the following simplified JSON?

      {
        "TableName": "PersonTable",
        "Key": {
          "id": {
            "N": "1"
          }
        },
        "ReturnValues": "ALL_NEW",
        "UpdateExpression": "SET #mainAddress.#city = :mainAddress_city",
```

```
        "ExpressionAttributeNames": {
          "#city": "city",
          "#mainAddress": "mainAddress"
        },
        "ExpressionAttributeValues": {
          ":mainAddress_city": {
            "S": "YourCity"
          }
        }
      }

However, assume that we don't know if the 'mainAddress' already exists. If it doesn't
 exist, the SDK would try to update
an attribute of a non-existent map, which results in an exception.

In this particular case, we would likely select option B (SCALAR_ONLY) to retain the
 other values of the 'mainAddress'.
*/
```

I due esempi seguenti mostrano gli usi di e dei valori enumeratiMAPS_ONLY. SCALAR_ONLY MAPS_ONLYaggiunge una mappa e SCALAR_ONLY aggiorna una mappa.

IgnoreNullsMode.MAPS_ONLY Esempio

```
    public void mapsOnlyModeExample() {
        // Assume that we ran this code yesterday.
        Person person = new Person();
        person.setId(1);
        person.setFirstName("FirstName");

        personDynamoDbTable.putItem(person);

        // Assume that we are running this code now.

        /* Note that we create a new Person with only the necessary information to
 update the city value
        of the mainAddress. */
        Person personForUpdate = new Person();
        personForUpdate.setId(1);
        // The update we want to make changes the city.
        Address mainAddressForUpdate = new Address();
        mainAddressForUpdate.setCity("YourCity");
        personForUpdate.setMainAddress(mainAddressForUpdate);
```

```
        Person updatedPerson = personDynamoDbTable.updateItem(r -> r
                .item(personForUpdate)
                .ignoreNullsMode(IgnoreNullsMode.MAPS_ONLY)); // Since the mainAddress
property does not exist, use MAPS_ONLY mode.
        assert updatedPerson.getMainAddress().getCity().equals("YourCity");
        assert updatedPerson.getMainAddress().getState() == null;
    }
```

IgnoreNullsMode.SCALAR_ONLY example

```
    public void scalarOnlyExample() {
        // Assume that we ran this code yesterday.
        Person person = new Person();
        person.setId(1);
        Address mainAddress = new Address();
        mainAddress.setCity("MyCity");
        mainAddress.setState("MyState");
        person.setMainAddress(mainAddress);

        personDynamoDbTable.putItem(person);

        // Assume that we are running this code now.

        /* Note that we create a new Person with only the necessary information to
update the city value
        of the mainAddress. */
        Person personForUpdate = new Person();
        personForUpdate.setId(1);
        // The update we want to make changes the city.
        Address mainAddressForUpdate = new Address();
        mainAddressForUpdate.setCity("YourCity");
        personForUpdate.setMainAddress(mainAddressForUpdate);

        Person updatedPerson = personDynamoDbTable.updateItem(r -> r
                .item(personForUpdate)
                .ignoreNullsMode(IgnoreNullsMode.SCALAR_ONLY)); // SCALAR_ONLY mode
ignores null properties in the in mainAddress.
        assert updatedPerson.getMainAddress().getCity().equals("YourCity");
        assert updatedPerson.getMainAddress().getState().equals("MyState"); // The
state property remains the same.
    }
```

Fate riferimento alla tabella seguente per vedere quali valori nulli vengono ignorati per ciascuna modalità. Spesso è possibile lavorare con entrambi`SCALAR_ONLY`, `MAPS_ONLY` tranne quando si lavora con bean o mappe.

Quali proprietà con valori nulli nell'oggetto inviato vengono ignorate dall'SDK per **`updateItem`** ciascuna modalità?

Tipo di proprietà	in modalità SCALAR_ONLY	in modalità MAPS_ONLY	in modalità DEFAULT
Scalare superiore	Sì	Sì	No
Bean o mappa	Sì	Sì	No
Valore scalare di un bean o di una voce di mappa	Sì[1]	N. 2	No
Elenco o set	Sì	Sì	No

[1] Ciò presuppone che la mappa esista già in DynamoDB. Qualsiasi valore scalare, nullo o non nullo, del bean o della mappa fornito nell'oggetto per l'aggiornamento richiede che esista un percorso al valore in DynamoDB. L'SDK costruisce un percorso verso l'attributo utilizzando l'operatore di dereferenza prima di inviare la richiesta. `. (dot)`

[2] Poiché si utilizza la `MAPS_ONLY` modalità per sostituire completamente o aggiungere un bean o una map, tutti i valori null nel bean o nella mappa vengono mantenuti nella mappa salvata in DynamoDB.

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 gettergetInnerBeanWithoutAnno(), 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 getter`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`istanza 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(NUL=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 per 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'`DynamoDbEnhancedClient`istanza, 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 DocumentTableSchemarichiede 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'`EnhancedDocument`istanza 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'`EnhancedDocument`istanza 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'`EnhancedDocument`istanza, 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 DynamoDbEnhancedClientdi 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 getAddressesForCustomMappingExample() 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 = getAddressesForCustomMappingExample();

        // 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'}
```

Codice **CustomAttributeConverterProvider**

```
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 oject.
        @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 an `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 il`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'`version`attributo 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())
```

AutoGeneratedUuidExtension

È possibile generare un UUID (Universally Unique Identifier) univoco per un attributo quando un nuovo record viene scritto nel database utilizzando. `AutoGeneratedUuidExtension` Il metodo Java JDK UUID.randomUUID () genera il valore e l'estensione viene applicata a un attributo di tipo. `java.lang.String`

Poiché Java SDK non carica questa estensione per impostazione predefinita, è necessario specificarla come estensione personalizzata quando si crea il client avanzato, come illustrato nel primo esempio di questo argomento.

L'`uniqueId`attributo è l'obiettivo del comportamento dell'estensione nel frammento seguente.

```
    @AutoGeneratedUuidExtension
    public String getUniqueId() {...}
    public void setUniqueId(String uniqueId) {...}
```

L'approccio equivalente allo schema di tabella statica è illustrato nel frammento seguente.

```
     .addAttribute(String.class, a -> a.name("uniqueId")
                                       .getter(Customer::getUniqueId)
                                       .setter(Customer::setUniqueId)
                                       // Applying the 'autoGeneratedUuid' tag to the
attribute.

.tags(AutoGeneratedUuidExtension.AttributeTags.autoGeneratedUuidAttribute())
```

Se desideri che l'estensione compili l'UUID solo per i `putItem` metodi e non per i `updateItem` metodi, aggiungi l'annotazione sul comportamento di aggiornamento come mostrato nel seguente frammento.

```
    @AutoGeneratedUuidExtension
    @DynamoDbUpdateBehavior(UpdateBehavior.WRITE_IF_NOT_EXISTS)
    public String getUniqueId() {...}
    public void setUniqueId(String uniqueId) {...}
```

Se utilizzate l'approccio dello schema a tabella statica, utilizzate il seguente codice equivalente.

```
     .addAttribute(String.class, a -> a.name("uniqueId")
                                       .getter(Customer::getUniqueId)
```

```
                                        .setter(Customer::setUniqueId)
                                        // Applying the 'autoGeneratedUuid' tag to the
attribute.

.tags(AutoGeneratedUuidExtension.AttributeTags.autoGeneratedUuidAttribute(),

StaticAttributeTags.updateBehavior(UpdateBehavior.WRITE_IF_NOT_EXISTS))
```

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 SdkPublisheranziché un valore restituito dalla modalità sincrona per SdkIterablegli 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
DynamoDbAtomicCounter	attributo [2]	Incrementa un attributo numerico con tag ogni volta che un record viene scritto nel database.	Introduzione e discussione.
DynamoDbAttribute	attributo	Definisce o rinomina una proprietà del bean mappata a un attributo di tabella DynamoDB.	• Discussione iniziale. • Sezione introduttiva: vedi Nota. • Esempi del metodo In Query in MovieActor classe.
DynamoDbAutoGeneratedTimestampAttribute	attributo	Aggiorna un attributo taggato con un timestamp corrente ogni volta che l'elemento viene scritto correttamente nel database	Introduzione e discussione.

Nome dell'annotazione	L'annotazione si applica a 1	Cosa fa	Dove viene mostrato in questa guida
DynamoDbAutoGeneratedUuid	attributo	Genera un UUID (Universally Unique Identifier) univoco per un attributo quando un nuovo record viene scritto nel database.	Introduzione e discussione.
DynamoDbBean	classe	Contrassegna una classe di dati come mappabile su uno schema tabellare.	Primo utilizzo sulla classe Customer nella sezione Guida introduttiva. In tutta la guida sono presenti diversi utilizzi.
DynamoDbConvertedBy	attributo	Associa un attributo personalizzato all'`AttributeConverter` attributo annotato.	Discussione iniziale ed esempio.
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.	• Discussione iniziale. • Implicazioni per altro codice.
DynamoDbIgnore	attributo	Fa sì che l'attributo rimanga non mappato.	• Discussione iniziale. • Uso in ProductCatalog classe.

Nome dell'annotazione	L'annotazione si applica a 1	Cosa fa	Dove viene mostrato in questa guida
DynamoDbIgnoreNulls	attributo	Impedisce il salvataggio degli attributi nulli degli oggetti annidati. DynamoDb	Discussioni ed esempi.
DynamoDbImmutable	classe	Contrassegna una classe di dati immutabile come mappabile su uno schema tabellare.	• Introduzione all'annotazione. • Uso in ProductCatalog classe. • Da usare con Lombok.
DynamoDbPartitionKey	attributo	Contrassegna un attributo come chiave di partizione primaria (chiave hash) della tabella. DynamoDb	• Utilizzo iniziale della classe Customer nella sezione Guida introduttiva. • Con Lombok.
DynamoDbPreserveEmptyObject	attributo	Speciifica che se non sono presenti dati per l'oggetto mappato all'attributo annotato, l'oggetto deve essere inizializzato con tutti i campi nulli.	Discussione ed esempi.

Nome dell'annotazione	L'annotazione si applica a 1	Cosa fa	Dove viene mostrato in questa guida
DynamoDbSecondaryPartitionKey	attributo	Contrassegna un attributo come chiave di partizione per un indice secondario globale.	• Utilizza negli indici secondari ed esempi. • Negli esempi del metodo Query. • Nell'esempio di Lombok • Con classi immutabili.
DynamoDbSecondarySortKey	attributo	Contrassegna un attributo come chiave di ordinamento opzionale per un indice secondario globale o locale.	• Utilizza negli indici secondari ed esempi. • Negli esempi del metodo Query. • Nell'esempio di Lombok. • Con classi immutabili.
DynamoDbSortKey	attributo	Contrassegna un attributo come chiave di ordinamento primaria opzionale (chiave di intervallo).	• Sezione introduttiva sulla classe Customer. • Con classi immutabili. • Nell'esempio di Lombok. • Negli esempi del metodo Query.

Nome dell'annotazione	L'annotazione si applica a 1	Cosa fa	Dove viene mostrato in questa guida
DynamoDbUpdateBehavior	attributo	Specifica il comportamento quando questo attributo viene aggiornato come parte di un'operazione di 'aggiornamento' come UpdateItem.	Introduzione ed esempio.
DynamoDbVersionAttribute	attributo	Incrementa il numero di versione di un articolo.	Introduzione e discussione.

[1] 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.

[2] 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 EC2che utilizzano il AWS SDK per 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
- Lavora con i metadati delle EC2 istanze Amazon

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) RunInstancesRequestcontenente 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 StartInstancesRequestcontenente 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 EC2 istanza, chiama il `stopInstances`metodo di Ec2Client, fornendogli un file StopInstancesRequestcontenente 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 EC2 istanza, chiama il `rebootInstances`metodo di Ec2Client, fornendogli un file RebootInstancesRequestcontenente 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 DescribeInstancesRequeste chiama il metodo di Ec2Client. `describeInstances` Restituirà un DescribeInstancesResponseoggetto 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

È possibile monitorare vari aspetti delle Amazon EC2 istanze, come l'utilizzo della CPU e della rete, la memoria disponibile e lo spazio rimanente su disco. 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'MonitorInstancesRequestcon 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 UnmonitorInstancesRequestcon 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

- RunInstancesnell' Amazon EC2 API Reference
- DescribeInstancesnell' Amazon EC2 API Reference
- StartInstancesnell' Amazon EC2 API Reference
- StopInstancesnell' Amazon EC2 API Reference
- RebootInstancesnell' Amazon EC2 API Reference
- MonitorInstancesnell' Amazon EC2 API Reference
- UnmonitorInstancesnell' Amazon EC2 API 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.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2AsyncClient;
import software.amazon.awssdk.services.ec2.model.DescribeRegionsResponse;
import software.amazon.awssdk.services.ec2.model.DescribeAvailabilityZonesResponse;
import java.util.concurrent.CompletableFuture;
```

Codice

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2AsyncClient;
import software.amazon.awssdk.services.ec2.model.DescribeRegionsResponse;
import software.amazon.awssdk.services.ec2.model.DescribeAvailabilityZonesResponse;
import java.util.concurrent.CompletableFuture;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DescribeRegionsAndZones {
    public static void main(String[] args) {
        Ec2AsyncClient ec2AsyncClient = Ec2AsyncClient.builder()
            .region(Region.US_EAST_1)
            .build();

        try {
            CompletableFuture<Void> future =
 describeEC2RegionsAndZonesAsync(ec2AsyncClient);
            future.join(); // Wait for both async operations to complete.
        } catch (RuntimeException rte) {
            System.err.println("An exception occurred: " + (rte.getCause() != null ?
 rte.getCause().getMessage() : rte.getMessage()));
        }
    }

    /**
     * Asynchronously describes the EC2 regions and availability zones.
     *
     * @param ec2AsyncClient the EC2 async client used to make the API calls
     * @return a {@link CompletableFuture} that completes when both the region and
 availability zone descriptions are complete
     */
    public static CompletableFuture<Void>
 describeEC2RegionsAndZonesAsync(Ec2AsyncClient ec2AsyncClient) {
        // Initiate the asynchronous request to describe regions
        CompletableFuture<DescribeRegionsResponse> regionsResponse =
 ec2AsyncClient.describeRegions();
```

```
        // Handle the response or exception for regions
        CompletableFuture<DescribeRegionsResponse> regionsFuture =
regionsResponse.whenComplete((regionsResp, ex) -> {
            if (ex != null) {
                // Handle the exception by throwing a RuntimeException
                throw new RuntimeException("Failed to describe EC2 regions.", ex);
            } else if (regionsResp == null || regionsResp.regions().isEmpty()) {
                // Throw an exception if the response is null or the result is empty
                throw new RuntimeException("No EC2 regions found.");
            } else {
                // Process the response if no exception occurred and the result is not
empty
                regionsResp.regions().forEach(region -> {
                    System.out.printf(
                        "Found Region %s with endpoint %s%n",
                        region.regionName(),
                        region.endpoint());
                });
            }
        });

        CompletableFuture<DescribeAvailabilityZonesResponse> zonesResponse =
ec2AsyncClient.describeAvailabilityZones();
        CompletableFuture<DescribeAvailabilityZonesResponse> zonesFuture =
zonesResponse.whenComplete((zonesResp, ex) -> {
            if (ex != null) {
                throw new RuntimeException("Failed to describe EC2 availability
zones.", ex);
            } else if (zonesResp == null || zonesResp.availabilityZones().isEmpty()) {
                throw new RuntimeException("No EC2 availability zones found.");
            } else {
                zonesResp.availabilityZones().forEach(zone -> {
                    System.out.printf(
                        "Found Availability Zone %s with status %s in region %s%n",
                        zone.zoneName(),
                        zone.state(),
                        zone.regionName()
                    );
                });
            }
        });

        return CompletableFuture.allOf(regionsFuture, zonesFuture);
```

```
    }
}
```

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 AvailabilityZoneoggetti che rappresentano ogni zona di disponibilità.

Importazioni

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2AsyncClient;
import software.amazon.awssdk.services.ec2.model.DescribeRegionsResponse;
import software.amazon.awssdk.services.ec2.model.DescribeAvailabilityZonesResponse;
import java.util.concurrent.CompletableFuture;
```

Codice

Crea Ec2Client.

```
        Ec2AsyncClient ec2AsyncClient = Ec2AsyncClient.builder()
            .region(Region.US_EAST_1)
            .build();
```

Quindi chiama describeAvailabilityZones () e recupera i risultati.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2AsyncClient;
import software.amazon.awssdk.services.ec2.model.DescribeRegionsResponse;
import software.amazon.awssdk.services.ec2.model.DescribeAvailabilityZonesResponse;
import java.util.concurrent.CompletableFuture;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
```

```
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DescribeRegionsAndZones {
    public static void main(String[] args) {
        Ec2AsyncClient ec2AsyncClient = Ec2AsyncClient.builder()
            .region(Region.US_EAST_1)
            .build();

        try {
            CompletableFuture<Void> future =
 describeEC2RegionsAndZonesAsync(ec2AsyncClient);
            future.join(); // Wait for both async operations to complete.
        } catch (RuntimeException rte) {
            System.err.println("An exception occurred: " + (rte.getCause() != null ?
 rte.getCause().getMessage() : rte.getMessage()));
        }
    }

    /**
     * Asynchronously describes the EC2 regions and availability zones.
     *
     * @param ec2AsyncClient the EC2 async client used to make the API calls
     * @return a {@link CompletableFuture} that completes when both the region and
 availability zone descriptions are complete
     */
    public static CompletableFuture<Void>
 describeEC2RegionsAndZonesAsync(Ec2AsyncClient ec2AsyncClient) {
        // Initiate the asynchronous request to describe regions
        CompletableFuture<DescribeRegionsResponse> regionsResponse =
 ec2AsyncClient.describeRegions();

        // Handle the response or exception for regions
        CompletableFuture<DescribeRegionsResponse> regionsFuture =
 regionsResponse.whenComplete((regionsResp, ex) -> {
            if (ex != null) {
                // Handle the exception by throwing a RuntimeException
                throw new RuntimeException("Failed to describe EC2 regions.", ex);
            } else if (regionsResp == null || regionsResp.regions().isEmpty()) {
                // Throw an exception if the response is null or the result is empty
                throw new RuntimeException("No EC2 regions found.");
            } else {
                // Process the response if no exception occurred and the result is not
 empty
                regionsResp.regions().forEach(region -> {
```

```
                    System.out.printf(
                        "Found Region %s with endpoint %s%n",
                        region.regionName(),
                        region.endpoint());
                });
            }
        });

        CompletableFuture<DescribeAvailabilityZonesResponse> zonesResponse =
 ec2AsyncClient.describeAvailabilityZones();
        CompletableFuture<DescribeAvailabilityZonesResponse> zonesFuture =
 zonesResponse.whenComplete((zonesResp, ex) -> {
            if (ex != null) {
                throw new RuntimeException("Failed to describe EC2 availability
 zones.", ex);
            } else if (zonesResp == null || zonesResp.availabilityZones().isEmpty()) {
                throw new RuntimeException("No EC2 availability zones found.");
            } else {
                zonesResp.availabilityZones().forEach(zone -> {
                    System.out.printf(
                        "Found Availability Zone %s with status %s in region %s%n",
                        zone.zoneName(),
                        zone.state(),
                        zone.regionName()
                    );
                });
            }
        });

        return CompletableFuture.allOf(regionsFuture, zonesFuture);
    }
}
```

Vedi l'esempio completo su. GitHub

Descrivere gli account

Per elencare le informazioni EC2 relative al 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 AccountAttributeoggetti. 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 AccountAttributeValueoggetti. 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.Ec2AsyncClient;
import software.amazon.awssdk.services.ec2.model.DescribeAccountAttributesResponse;
import java.util.concurrent.CompletableFuture;
```

Codice

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2AsyncClient;
import software.amazon.awssdk.services.ec2.model.DescribeAccountAttributesResponse;
import java.util.concurrent.CompletableFuture;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * 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) {
        Ec2AsyncClient ec2AsyncClient = Ec2AsyncClient.builder()
            .region(Region.US_EAST_1)
            .build();

        try {
            CompletableFuture<DescribeAccountAttributesResponse> future =
 describeEC2AccountAsync(ec2AsyncClient);
            future.join();
            System.out.println("EC2 Account attributes described successfully.");
        } catch (RuntimeException rte) {
            System.err.println("An exception occurred: " + (rte.getCause() != null ?
 rte.getCause().getMessage() : rte.getMessage()));
        }
    }
```

```
    /**
     * Describes the EC2 account attributes asynchronously.
     *
     * @param ec2AsyncClient the EC2 asynchronous client to use for the operation
     * @return a {@link CompletableFuture} containing the {@link
 DescribeAccountAttributesResponse} with the account attributes
     */
    public static CompletableFuture<DescribeAccountAttributesResponse>
 describeEC2AccountAsync(Ec2AsyncClient ec2AsyncClient) {
        CompletableFuture<DescribeAccountAttributesResponse> response =
 ec2AsyncClient.describeAccountAttributes();
        return response.whenComplete((accountResults, ex) -> {
            if (ex != null) {
                // Handle the exception by throwing a RuntimeException.
                throw new RuntimeException("Failed to describe EC2 account
 attributes.", ex);
            } else if (accountResults == null ||
 accountResults.accountAttributes().isEmpty()) {
                // Throw an exception if the response is null or no account attributes
 are found.
                throw new RuntimeException("No account attributes found.");
            } else {
                // Process the response if no exception occurred.
                accountResults.accountAttributes().forEach(attribute -> {
                    System.out.println("\nThe name of the attribute is " +
 attribute.attributeName());
                    attribute.attributeValues().forEach(
                        myValue -> System.out.println("The value of the attribute is "
 + myValue.attributeValue()));
                });
            }
        });
    }
}
```

Vedi l'esempio completo su GitHub.

Ulteriori informazioni

- Regioni e zone di disponibilità nella Guida per l' Amazon EC2 utente per le istanze Linux
- DescribeRegionsnel riferimento alle Amazon EC2 API
- DescribeAvailabilityZonesnell' 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 CreateSecurityGroupRequestche 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 AuthorizeSecurityGroupEgressRequestEc2Client. `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 DescribeSecurityGroupsResponsecomando che puoi usare per accedere all'elenco dei gruppi di sicurezza chiamando il relativo `securityGroups` metodo, che restituisce un elenco di SecurityGroupoggetti.

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 DeleteSecurityGroupRequestche 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
- CreateSecurityGroupnella guida di riferimento alle API Amazon EC2
- DescribeSecurityGroupsnell' Amazon EC2 API Reference
- DeleteSecurityGroupnell' Amazon EC2 API Reference
- AuthorizeSecurityGroupIngressnell' Amazon EC2 API Reference

Lavora con i metadati delle EC2 istanze Amazon

Un client Java SDK per Amazon EC2 Instance Metadata Service (client di metadati) consente alle applicazioni di accedere ai metadati sulla loro istanza locale. EC2 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 di client. Il sincrono `Ec2MetadataClient` serve per bloccare le operazioni e l'altro per i casi d'uso asincroni e `Ec2MetadataAsyncClient`non bloccanti.

Inizia a usare

Per utilizzare il client di metadati, aggiungi l'artefatto Maven al tuo progetto`imds`. Sono inoltre necessarie classi per un `SdkHttpClient` (o un `SdkAsyncHttpClient` per la variante asincrona) sul classpath.

Il seguente codice XML Maven mostra i frammenti di dipendenza per l'utilizzo del sincrono insieme alla dipendenza per i client di metadati. UrlConnectionHttpClient

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

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

Cerca nel repository centrale di Maven la versione più recente dell'artefatto. `bom`

Per utilizzare un client HTTP asincrono, sostituisci lo snippet di dipendenza per l'artefatto. `url-connection-client` Ad esempio, il seguente frammento introduce l'implementazione. NettyNioAsyncHttpClient

```
    <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. A tale scopo, chiamate il `Ec2MetadataClient#create()` metodo statico come illustrato nel frammento seguente.

```
Ec2MetadataClient client = Ec2MetadataClient.create(); //
 'Ec2MetadataAsyncClient#create' is the asynchronous version.
```

Se l'applicazione ha più implementazioni dell'`SdkHttpAsyncClient`interfaccia `SdkHttpClient` or, è necessario specificare un'implementazione da utilizzare per il client di metadati, come illustrato nella sezione. the section called “Client HTTP configurabile”

> **Note**
>
> Per la maggior parte dei client di servizi, come Amazon S3, l'SDK for Java aggiunge automaticamente implementazioni dell'interfaccia or. `SdkHttpClient` `SdkHttpAsyncClient` Se il tuo client di metadati utilizza la stessa implementazione, funzionerà. `Ec2MetadataClient#create()` Se hai bisogno di un'implementazione diversa, devi specificarla quando crei il client di metadati.

Inviare 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 di metadati dell'istanza.

L'esempio seguente stampa il valore associato alla chiave sulla console. `ami-id`

```
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.

Riutilizzate la stessa istanza del client per più richieste, ma chiamate `close` il client quando non è più necessario per rilasciare risorse. Dopo la chiamata del metodo close, l'istanza del client non può più essere utilizzata.

Analizza le risposte

EC2 i metadati delle istanze possono essere emessi in diversi formati. Il testo normale e JSON sono i formati più utilizzati. I client di metadati offrono modi per lavorare con questi formati.

Come illustrato nell'esempio seguente, utilizzate il `asString` metodo per ottenere i dati come stringa Java. È inoltre possibile utilizzare il `asList` metodo per separare una risposta in testo semplice 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, utilizzate 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 esaminare la risposta in modo più dettagliato. Consulta la tabella delle categorie di metadati delle istanze per scoprire quali categorie di metadati forniscono risposte in formato JSON.

Configura 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 hanno esito negativo per motivi imprevisti. Per

impostazione predefinita, il client riprova tre volte in caso di richiesta non riuscita, con un tempo di backoff esponenziale tra un tentativo e l'altro.

Se il tuo caso d'uso richiede un meccanismo di ripetizione dei tentativi diverso, puoi personalizzare il client utilizzando il metodo del relativo generatore. `retryPolicy` Ad esempio, l'esempio seguente mostra un client sincrono configurato con un ritardo fisso di due secondi tra i tentativi e cinque tentativi di nuovo tentativo.

```
BackoffStrategy fixedBackoffStrategy =
 FixedDelayBackoffStrategy.create(Duration.ofSeconds(2));
Ec2MetadataClient client =
    Ec2MetadataClient.builder()
                     .retryPolicy(retryPolicyBuilder ->
 retryPolicyBuilder.numRetries(5)

 .backoffStrategy(fixedBackoffStrategy))
                     .build();
```

Ce ne sono diversi BackoffStrategiesche è possibile utilizzare con un client di metadati.

Puoi anche disabilitare completamente il meccanismo di riprova, come mostra il seguente frammento.

```
Ec2MetadataClient client =
    Ec2MetadataClient.builder()
                    .retryPolicy(Ec2MetadataRetryPolicy.none())
                    .build();
```

L'utilizzo `Ec2MetadataRetryPolicy#none()` disattiva la politica di riprova predefinita in modo che il client di metadati non tenti di riprovare.

Versione IP

Per impostazione predefinita, un client di metadati utilizza l' IPV4 endpoint in. `http://169.254.169.254` Per modificare il client in modo che utilizzi la IPV6 versione, usa il metodo `endpointMode` o il `endpoint` metodo del generatore. Si verifica un'eccezione se entrambi i metodi vengono chiamati sul builder.

Gli esempi seguenti mostrano entrambe le IPV6 opzioni.

```
Ec2MetadataClient client =
    Ec2MetadataClient.builder()
```

```
                    .endpointMode(EndpointMode.IPV6)
                    .build();
```

```
Ec2MetadataClient client =
    Ec2MetadataClient.builder()
                    .endpoint(URI.create("http://[fd00:ec2::254]"))
                    .build();
```

Funzionalità principali

Client asincrono

Per utilizzare la versione non bloccante del client, crea un'istanza della classe. `Ec2MetadataAsyncClient` Il codice nell'esempio seguente crea un client asincrono con impostazioni predefinite e utilizza il `get` metodo per recuperare il valore della chiave. `ami-id`

```
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 i `ami-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 puoi utilizzare per fornire un client HTTP personalizzato.

L'esempio seguente mostra il codice per un'istanza personalizzata`UrlConnectionHttpClient`.

```
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 utilizzano i metadati IMDSv2, tutte le richieste sono associate a una sessione. Una sessione è definita da un token con scadenza, che il client di metadati gestisce per voi. Ogni richiesta di metadati riutilizza automaticamente il token fino alla scadenza.

Per impostazione predefinita, un token dura sei ore (21.600 secondi). Ti consigliamo di mantenere il time-to-live valore predefinito, a meno che il tuo caso d'uso specifico non richieda una configurazione avanzata.

Se necessario, configura la durata utilizzando il metodo `tokenTtl` builder. Ad esempio, il codice nel frammento seguente crea un client con una durata della sessione di cinque minuti.

```
Ec2MetadataClient client =
    Ec2MetadataClient.builder()
                     .tokenTtl(Duration.ofMinutes(5))
                     .build();
```

Se omettete di chiamare il `tokenTtl` metodo sul builder, viene invece utilizzata la durata predefinita di 21.600.

Lavora con IAM

Questa sezione fornisce esempi di programmazione AWS Identity and Access Management (IAM) utilizzando la versione AWS SDK per 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

- Gestire le 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

Gestire le 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_GLOBALaffinché `IamClient` le chiamate funzionino perché IAM si tratta di 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 "";
    }
```

Vedi 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 `alistAccessKeys`, 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

- CreateAccessKeynell' IAM API Reference
- ListAccessKeysnell' IAM API Reference
- GetAccessKeyLastUsednell' IAM API Reference
- UpdateAccessKeynell' IAM API Reference
- DeleteAccessKeynell' 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 CreateUserRequestoggetto 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'ListUsersResponseoggetto 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 UpdateUserRequestoggetto 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 UpdateUserRequestoggetto 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

- IAM Utenti nella Guida per l' IAM utente
- Gestione IAM degli utenti nella Guida IAM per l'utente
- CreateUsernel riferimento alle IAM API
- ListUsersnell' IAM API Reference
- UpdateUsernell' IAM API Reference

- DeleteUsernell' IAM API 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>2.27.21</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:::amzn-s3-demo-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:::amzn-s3-demo-bucket/*",
    "Condition" : {
      "StringEquals" : {
        "s3:x-amz-acl" : "bucket-owner-full-control"
      }
    }
  }
}
```

Usa un **`IamPolicy`** con IAM

Dopo aver creato un'`IamPolicy`istanza, usi an `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 CreatePolicyRequestmetodo 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 GetPolicyRequestdi 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 ListAttachedRolePoliciesRequestoggetto che contenga il nome del ruolo per cui elencare le politiche.

Richiama `getAttachedPolicies` l'ListAttachedRolePoliciesResponseoggetto 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' IAM utente.
- AWS Riferimento alla politica IAM nella Guida IAM per l'utente.
- CreatePolicynell' IAM API Reference
- GetPolicynell' IAM API Reference

- AttachRolePolicynell' IAM API Reference
- ListAttachedRolePoliciesnell' IAM API Reference
- DetachRolePolicynell' IAM API 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 GetServerCertificateRequestcon 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.serverCertificate().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 ServerCertificateMetadataoggetti 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 UpdateServerCertificateRequestoggetto 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 DeleteServerCertificateRequestcontenente 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
- GetServerCertificatenell' IAM API Reference
- ListServerCertificatesnell' IAM API Reference
- UpdateServerCertificatenell' IAM API Reference
- DeleteServerCertificatenell' IAM API Reference
- AWS Certificate Manager Guida per l'utente

Lavora con Kinesis

Questa sezione fornisce esempi di programmazione Amazon Kinesiscon AWS SDK per 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

- Iscriviti a Amazon Kinesis Data Streams

Iscriviti a 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, SubscribeToShardRequestcrea un client asincrono e un oggetto. Kinesis Questi oggetti vengono utilizzati in ciascuno degli esempi seguenti per sottoscrivere gli 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 Builder

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 on`EventStream`, 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'KinesisClientoggetto per scrivere record di dati in un flusso di Kinesis dati utilizzando il `putRecords` metodo. Per richiamare correttamente questo metodo, create un PutRecordsRequestoggetto. 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'StockTradeoggetto 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>2.2.21</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

- SubscribeToShardEventnell' Amazon Kinesis API Reference
- SubscribeToShardnell' Amazon Kinesis API Reference

Funzioni di richiamo, elenco ed eliminazione AWS Lambda

Questa sezione fornisce esempi di programmazione con il client di Lambda servizio utilizzando AWS SDK per 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 informazioni di base per lavorare con Amazon S3 utilizzando. AWS SDK for Java 2.x Questa sezione completa gli esempi di Amazon S3 Java v2 presentati nella sezione Esempi di codice di questa guida.

Client S3 in AWS SDK for Java 2.x

AWS SDK for Java 2.x Fornisce diversi tipi di client S3. La tabella seguente mostra le differenze e può aiutarti a decidere qual è la soluzione migliore per i tuoi casi d'uso.

Diverse versioni di client Amazon S3

Client S3	Breve descrizione	Quando usare	Limitazione/inconveniente
AWS Client S3 basato su CRT Interfaccia: S3 AsyncClient Costruttore: S3 CrtAsyncClientBuilder	• Fornisce le stesse operazioni API asincrone del client asincrono S3 basato su Java ma con prestazioni superiori.	• La tua applicazione trasferisce oggetti di grandi dimensioni (> 8 MB) e desideri prestazioni massimizzate. • Desiderate caricare oggetti con	• Supporta un numero inferiore di impostazioni di configurazione rispetto ai client S3 basati su Java.

Client S3	Breve descrizione	Quando usare	Limitazione/inconveniente
	• `aws-crt`Richiede la dipendenza. • Supporta trasferimenti paralleli automatici (multiparte). Consultare the section called “Usa un client S3 performante”.	una lunghezza del contenuto sconosciuta. • Desideri un pool di connessioni e un bilanciamento del carico DNS avanzati, che migliorano la velocità di trasmissione e le prestazioni. • Desiderate una maggiore affidabilità del trasferimento in caso di guasto della rete. Le singole parti guaste vengono ritentate senza riavviare il trasferimento dall'inizio.	• Richiede una dipendenza aggiuntiva.

Client S3	Breve descrizione	Quando usare	Limitazione/inconveniente
Client asincrono S3 basato su Java con multipart abilitato Interfaccia: S3 AsyncClient Costruttore: S3 AsyncClientBuilder	• Fornisce un'API asincrona. • Supporta i trasferimenti paralleli automatici (multiparte) quando si abilita il multipart al momento della creazione. Consultare the section called “Configurare il supporto per il trasferimento parallelo”.	• La tua applicazione trasferisce oggetti di grandi dimensioni e desideri prestazioni migliorate. • vuoi caricare un oggetto con una lunghezza del contenuto sconosciuta. • Desiderate una maggiore affidabilità del trasferimento in caso di guasto della rete. Le singole parti guaste vengono ritentate senza riavviare il trasferimento dall'inizio. • Sono necessarie opzioni di configurazione che non sono disponibili con il client S3 basato su AWS CRT.	Meno performante del client S3 basato su CRT. AWS

Client S3	Breve descrizione	Quando usare	Limitazione/inconveniente
Client asincrono S3 basato su Java senza multipart abilitato Interfaccia: S3 AsyncClient Costruttore: S3 AsyncClientBuilder	• Fornisce un'API asincrona.	• Stai trasferendo oggetti che pesano meno di 8 MB. • Vuoi un'API asincrona.	Nessuna ottimizzazione delle prestazioni.
Client di sincronizzazione S3 basato su Java Interfaccia: S3Client Costruttore: S3 ClientBuilder	• Fornisce un'API sincrona.	• Stai trasferendo oggetti che pesano meno di 8 MB. • Vuoi un'API sincrona.	Nessuna ottimizzazione delle prestazioni.

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();
```

Argomenti

- Caricamento di stream su Amazon S3 utilizzando AWS SDK for Java 2.x
- Lavora con prodotti Amazon S3 prefirmati URLs
- Accesso interregionale per Amazon S3
- Protezione dell'integrità dei dati con checksum
- Usa un client S3 performante: client S3 basato su AWS CRT
- Configura il client asincrono S3 basato su Java per utilizzare trasferimenti paralleli
- Trasferisci file e directory con Amazon S3 Transfer Manager
- Lavora con le notifiche degli eventi S3

Caricamento di stream su Amazon S3 utilizzando AWS SDK for Java 2.x

Quando usi uno stream per caricare contenuti su S3 utilizzando `putObject`o `uploadPart`, utilizzi una classe di `RequestBody` fabbrica per l'API sincrona per fornire lo stream. Per l'API asincrona, `AsyncRequestBody` è la classe factory equivalente.

Quali metodi caricano gli stream?

Per l'API sincrona, puoi utilizzare i seguenti metodi di fabbrica `RequestBody` per fornire lo stream:

- `fromInputStream(InputStream inputStream, long contentLength)`

  `fromContentProvider(ContentStreamProvider provider, long contentLength, String mimeType)`
  - `ContentStreamProvider`Ha il metodo `fromInputStream(InputStream inputStream)` factory
- `fromContentProvider(ContentStreamProvider provider, String mimeType)`

Per l'API asincrona, puoi utilizzare i seguenti metodi di fabbrica di: `AsyncRequestBody`

- `fromInputStream(InputStream inputStream, Long contentLength, ExecutorService executor)`

- `fromInputStream(AsyncRequestBodyFromInputStreamConfiguration configuration)`
  - Si utilizza AsyncRequestBodyFromInputStreamConfiguration .Builder per fornire lo stream
- `fromInputStream(Consumer<AsyncRequestBodyFromInputStreamConfiguration.Builder> configuration)`
- `forBlockingInputStream(Long contentLength)`
  - Il risultato `BlockingInputStreamAsyncRequestBody` contiene il metodo `writeInputStream(InputStream inputStream)` che è possibile utilizzare per fornire lo stream

Esecuzione del caricamento

Se conosci la durata dello stream

Come puoi vedere dalla firma dei metodi mostrata in precedenza, la maggior parte dei metodi accetta un parametro di lunghezza del contenuto.

Se conosci la lunghezza del contenuto in byte, fornisci il valore esatto:

```
// Always provide the exact content length when it's available.
long contentLength = 1024; // Exact size in bytes.
s3Client.putObject(req -> req
    .bucket("my-bucket")
    .key("my-key"),
RequestBody.fromInputStream(inputStream, contentLength));
```

> **Warning**
>
> Quando carichi da un flusso di input, se la lunghezza del contenuto specificata non corrisponde al numero effettivo di byte, potresti riscontrare:
>
> - Oggetti troncati se la lunghezza specificata è troppo piccola
> - Caricamenti non riusciti o connessioni sospese se la lunghezza specificata è troppo grande

Se non conosci la durata dello stream

Utilizzando l'API sincrona

Usa: `fromContentProvider(ContentStreamProvider provider, String mimeType)`

```
public PutObjectResponse syncClient_stream_unknown_size(String bucketName, String key,
 InputStream inputStream) {

    S3Client s3Client = S3Client.create();

    RequestBody body =
 RequestBody.fromContentProvider(ContentStreamProvider.fromInputStream(inputStream),
 "text/plain");
    PutObjectResponse putObjectResponse = s3Client.putObject(b ->
 b.bucket(BUCKET_NAME).key(KEY_NAME), body);
    return putObjectResponse;
}
```

Poiché l'SDK memorizza nel buffer l'intero flusso per calcolare la lunghezza del contenuto, è possibile che si verifichino problemi di memoria con flussi di grandi dimensioni. Se devi caricare stream di grandi dimensioni con il client sincrono, prendi in considerazione l'utilizzo dell'API multipart:

Carica uno stream utilizzando l'API client sincrona e l'API multipart

```
public static void uploadStreamToS3(String bucketName, String key, InputStream
 inputStream) {
    // Create S3 client
    S3Client s3Client = S3Client.create();
    try {
        // Step 1: Initiate the multipart upload
        CreateMultipartUploadRequest createMultipartUploadRequest =
 CreateMultipartUploadRequest.builder()
                .bucket(bucketName)
                .key(key)
                .build();

        CreateMultipartUploadResponse createResponse =
 s3Client.createMultipartUpload(createMultipartUploadRequest);
        String uploadId = createResponse.uploadId();
        System.out.println("Started multipart upload with ID: " + uploadId);

        // Step 2: Upload parts
```

```
        List<CompletedPart> completedParts = new ArrayList<>();
        int partNumber = 1;
        byte[] buffer = new byte[PART_SIZE];
        int bytesRead;

        try {
            while ((bytesRead = readFullyOrToEnd(inputStream, buffer)) > 0) {
                // Create request to upload a part
                UploadPartRequest uploadPartRequest = UploadPartRequest.builder()
                        .bucket(bucketName)
                        .key(key)
                        .uploadId(uploadId)
                        .partNumber(partNumber)
                        .build();

                // If we didn't read a full buffer, create a properly sized byte array
                RequestBody requestBody;
                if (bytesRead < PART_SIZE) {
                    byte[] lastPartBuffer = new byte[bytesRead];
                    System.arraycopy(buffer, 0, lastPartBuffer, 0, bytesRead);
                    requestBody = RequestBody.fromBytes(lastPartBuffer);
                } else {
                    requestBody = RequestBody.fromBytes(buffer);
                }

                // Upload the part and save the response's ETag
                UploadPartResponse uploadPartResponse =
s3Client.uploadPart(uploadPartRequest, requestBody);
                CompletedPart part = CompletedPart.builder()
                        .partNumber(partNumber)
                        .eTag(uploadPartResponse.eTag())
                        .build();
                completedParts.add(part);

                System.out.println("Uploaded part " + partNumber + " with size " +
bytesRead + " bytes");
                partNumber++;
            }

            // Step 3: Complete the multipart upload
            CompletedMultipartUpload completedMultipartUpload =
CompletedMultipartUpload.builder()
                    .parts(completedParts)
                    .build();
```

```
            CompleteMultipartUploadRequest completeRequest =
 CompleteMultipartUploadRequest.builder()
                    .bucket(bucketName)
                    .key(key)
                    .uploadId(uploadId)
                    .multipartUpload(completedMultipartUpload)
                    .build();

            CompleteMultipartUploadResponse completeResponse =
 s3Client.completeMultipartUpload(completeRequest);
            System.out.println("Multipart upload completed. Object URL: " +
 completeResponse.location());

        } catch (Exception e) {
            // If an error occurs, abort the multipart upload
            System.err.println("Error during multipart upload: " + e.getMessage());
            AbortMultipartUploadRequest abortRequest =
 AbortMultipartUploadRequest.builder()
                    .bucket(bucketName)
                    .key(key)
                    .uploadId(uploadId)
                    .build();
            s3Client.abortMultipartUpload(abortRequest);
            System.err.println("Multipart upload aborted");
        } finally {
            try {
                inputStream.close();
            } catch (IOException e) {
                System.err.println("Error closing input stream: " + e.getMessage());
            }
        }
    } finally {
        s3Client.close();
    }
}

/**
 * Reads from the input stream into the buffer, attempting to fill the buffer
 completely
 * or until the end of the stream is reached.
 *
 * @param inputStream the input stream to read from
 * @param buffer      the buffer to fill
```

```
 * @return the number of bytes read, or -1 if the end of the stream is reached before
 any bytes are read
 * @throws IOException if an I/O error occurs
 */
private static int readFullyOrToEnd(InputStream inputStream, byte[] buffer) throws
 IOException {
    int totalBytesRead = 0;
    int bytesRead;

    while (totalBytesRead < buffer.length) {
        bytesRead = inputStream.read(buffer, totalBytesRead, buffer.length -
 totalBytesRead);
        if (bytesRead == -1) {
            break; // End of stream
        }
        totalBytesRead += bytesRead;
    }

    return totalBytesRead > 0 ? totalBytesRead : -1;
}
```

> **Note**
>
> Nella maggior parte dei casi d'uso, consigliamo di utilizzare l'API client asincrona per flussi di dimensioni sconosciute. Questo approccio consente i trasferimenti paralleli e offre un'interfaccia di programmazione più semplice, poiché l'SDK gestisce la segmentazione del flusso in blocchi multiparte se lo stream è di grandi dimensioni.
> Sia il client asincrono S3 standard con multipart abilitato sia il client S3 basato su CRT implementano questo approccio. AWS Mostriamo esempi di questo approccio nella sezione seguente.

Utilizzo dell'API asincrona

È possibile fornire `null` l'argomento al `contentLength` `fromInputStream(InputStream inputStream, Long contentLength, ExecutorService executor)`

Example utilizzando il client AWS asincrono basato su CRT:

```
public PutObjectResponse crtClient_stream_unknown_size(String bucketName, String key,
 InputStream inputStream) {
```

```
    S3AsyncClient s3AsyncClient = S3AsyncClient.crtCreate();
    ExecutorService executor = Executors.newSingleThreadExecutor();
    AsyncRequestBody body = AsyncRequestBody.fromInputStream(inputStream, null,
 executor);  // 'null' indicates that the

     // content length is unknown.
    CompletableFuture<PutObjectResponse> responseFuture =
            s3AsyncClient.putObject(r -> r.bucket(bucketName).key(key), body)
                    .exceptionally(e -> {
                        if (e != null){
                            logger.error(e.getMessage(), e);
                        }
                        return null;
                    });

    PutObjectResponse response = responseFuture.join(); // Wait for the response.
    executor.shutdown();
    return response;
}
```

Example utilizzando il client asincrono standard con multipart abilitato:

```
public PutObjectResponse asyncClient_multipart_stream_unknown_size(String bucketName,
 String key, InputStream inputStream) {

    S3AsyncClient s3AsyncClient =
 S3AsyncClient.builder().multipartEnabled(true).build();
    ExecutorService executor = Executors.newSingleThreadExecutor();
    AsyncRequestBody body = AsyncRequestBody.fromInputStream(inputStream, null,
 executor); // 'null' indicates that the

    // content length is unknown.
    CompletableFuture<PutObjectResponse> responseFuture =
            s3AsyncClient.putObject(r -> r.bucket(bucketName).key(key), body)
                    .exceptionally(e -> {
                        if (e != null) {
                            logger.error(e.getMessage(), e);
                        }
                        return null;
                    });

    PutObjectResponse response = responseFuture.join(); // Wait for the response.
```

```
    executor.shutdown();
    return response;
}
```

Lavora con prodotti Amazon S3 prefirmati URLs

Pre-signed URLs fornisce 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 richieste HTTP GET e HTTP PUT URLs prefirmate.

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 **`HttpURLConnection`** (dalla v1.1)

```
    /* Use the JDK HttpURLConnection (since v1.1) class to do the upload. */
    public void useHttpUrlConnectionToPut(String presignedUrlString, File fileToPut,
Map<String, String> metadata) {
        logger.info("Begin [{}] upload", fileToPut.toString());
        try {
            URL presignedUrl = new URL(presignedUrlString);
            HttpURLConnection connection = (HttpURLConnection)
presignedUrl.openConnection();
            connection.setDoOutput(true);
            metadata.forEach((k, v) -> connection.setRequestProperty("x-amz-meta-" + k,
v));
            connection.setRequestMethod("PUT");
            OutputStream out = connection.getOutputStream();

            try (RandomAccessFile file = new RandomAccessFile(fileToPut, "r");
                 FileChannel inChannel = file.getChannel()) {
                ByteBuffer buffer = ByteBuffer.allocate(8192); //Buffer size is 8k

                while (inChannel.read(buffer) > 0) {
                    buffer.flip();
                    for (int i = 0; i < buffer.limit(); i++) {
                        out.write(buffer.get());
                    }
                    buffer.clear();
                }
            } catch (IOException e) {
                logger.error(e.getMessage(), e);
            }

            out.close();
            connection.getResponseCode();
            logger.info("HTTP response code is " + connection.getResponseCode());

        } catch (S3Exception | IOException e) {
```

```
            logger.error(e.getMessage(), e);
        }
    }
```

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.

Configurazione

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>2.27.21</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.

Protezione dell'integrità dei dati con checksum

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 Guida per l'utente di Amazon Simple Storage Service, che include gli algoritmi supportati.

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 fornire un valore di checksum precalcolato utilizzando uno degli algoritmi supportati.

> **Note**
>
> L'SDK fornisce anche impostazioni globali per la protezione dell'integrità dei dati che puoi impostare esternamente, come puoi leggere nella Guida di riferimento agli strumenti.AWS SDKs

Discutiamo i checksum in due fasi di richiesta: caricamento di un oggetto e download di un oggetto.

Caricamento di un oggetto

Se non fornisci un algoritmo di checksum con la richiesta, il comportamento del checksum varia a seconda della versione dell'SDK che usi, come mostrato nella tabella seguente.

Comportamento del checksum quando non viene fornito alcun algoritmo di checksum

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 precalcolato. SHA256

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.

Donwload di un oggetto

La richiesta nel seguente frammento indica all'SDK di convalidare il checksum nella risposta calcolando il checksum e confrontando i valori.

> **Note**
>
> Se l'oggetto non è stato caricato con un checksum, non viene effettuata alcuna convalida.

Usa un client S3 performante: client S3 basato su AWS CRT

Il client S3 AWS basato su CRT, basato su AWS Common 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 multiparte 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.

AWS Componenti 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'SdkAsyncHttpClientinterfaccia 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 AWS Common 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>2.27.21</version>
</dependency>
<dependency>
  <groupId>software.amazon.awssdk.crt</groupId>
  <artifactId>aws-crt</artifactId>
  <version>0.30.11</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 PutObjecte GetObjectdisponibili tramite. AWS SDK per 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();
```

Caricamento di stream di dimensioni sconosciute

Un vantaggio significativo del client S3 AWS AWS basato su CRT è la sua capacità di gestire in modo efficiente flussi di input di dimensioni sconosciute. Ciò è particolarmente utile quando è necessario caricare dati da una fonte in cui la dimensione totale non può essere determinata in anticipo.

```
public PutObjectResponse crtClient_stream_unknown_size(String bucketName, String key,
 InputStream inputStream) {

    S3AsyncClient s3AsyncClient = S3AsyncClient.crtCreate();
    ExecutorService executor = Executors.newSingleThreadExecutor();
    AsyncRequestBody body = AsyncRequestBody.fromInputStream(inputStream, null,
 executor);  // 'null' indicates that the

     // content length is unknown.
    CompletableFuture<PutObjectResponse> responseFuture =
            s3AsyncClient.putObject(r -> r.bucket(bucketName).key(key), body)
                    .exceptionally(e -> {
                        if (e != null){
                            logger.error(e.getMessage(), e);
                        }
                        return null;
                    });

    PutObjectResponse response = responseFuture.join(); // Wait for the response.
    executor.shutdown();
    return response;
}
```

Questa funzionalità consente di evitare problemi comuni con i caricamenti tradizionali, in cui una specifica errata della lunghezza dei contenuti può portare a oggetti troncati o a caricamenti non riusciti.

Limitazioni di configurazione

Il client S3 AWS basato su CRT e il client asincrono S3 basato su Java offrono funzionalità comparabili, mentre il client S3 basato su CRT offre un vantaggio in termini di prestazioni. AWS Tuttavia, il client S3 basato su CRT non dispone delle impostazioni di configurazione del AWS client asincrono S3 basato su Java. Queste impostazioni includono:

- Configurazione a livello di client: timeout dei tentativi di chiamata API, intercettori di esecuzione della compressione, editori di metriche, attributi di esecuzione personalizzati, opzioni avanzate personalizzate, servizio esecutore pianificato personalizzato, intestazioni personalizzate
- Configurazione a livello di richiesta: firmatari personalizzati, fornitori di credenziali, timeout dei tentativi di chiamata API

Per un elenco completo delle differenze di configurazione, consulta il riferimento all'API.

Client asincrono S3 basato su Java	AWS Client S3 basato su CRT
Configurazioni a livello di client • ClientOverrideConfiguration.Builder	Configurazioni a livello di client • S3CrtAsyncClientBuilder
Configurazioni a livello di richiesta • RequestOverrideConfiguration.Builder • AwsRequestOverrideConfiguration.Costruttore	Nessuna configurazione a livello di richiesta

Configura il client asincrono S3 basato su Java per utilizzare trasferimenti paralleli

Dalla versione 2.27.5, il client asincrono S3 standard basato su Java supporta trasferimenti paralleli automatici (caricamenti e download multiparte). Il supporto per i trasferimenti paralleli viene configurato quando si crea il client asincrono S3 basato su Java.

Questa sezione mostra come abilitare i trasferimenti paralleli e come personalizzare la configurazione.

Creare un'istanza di **`S3AsyncClient`**

Quando crei un'`S3AsyncClient`istanza senza chiamare nessuno dei `multipart*` metodi sul builder, i trasferimenti paralleli non sono abilitati. Ciascuna delle seguenti istruzioni crea un client asincrono S3 basato su Java senza supporto per caricamenti e download multiparte.

Crea senza supporto multiparte

Example

```
import software.amazon.awssdk.auth.credentials.ProcessCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3AsyncClient;


S3AsyncClient s3Client = S3AsyncClient.create();

S3AsyncClient s3Client2 = S3AsyncClient.builder().build();

S3AsyncClient s3Client3 = S3AsyncClient.builder()
        .credentialsProvider(ProcessCredentialsProvider.builder().build())
        .region(Region.EU_NORTH_1)
        .build();
```

Crea con il supporto multiparte

Per abilitare i trasferimenti paralleli con le impostazioni predefinite, chiama `multipartEnabled` on the builder e passali `true` come mostrato nell'esempio seguente.

Example

```
S3AsyncClient s3AsyncClient2 = S3AsyncClient.builder()
        .multipartEnabled(true)
        .build();
```

Il valore predefinito è 8 MiB per `thresholdInBytes` e `minimumPartSizeInBytes` impostazioni.

Se si personalizzano le impostazioni multiparte, i trasferimenti paralleli vengono abilitati automaticamente come illustrato di seguito.

Example

```
import software.amazon.awssdk.services.s3.S3AsyncClient;
import static software.amazon.awssdk.transfer.s3.SizeConstant.MB;


S3AsyncClient s3AsyncClient2 = S3AsyncClient.builder()
        .multipartConfiguration(b -> b
                .thresholdInBytes(16 * MB)
```

```
                .minimumPartSizeInBytes(10 * MB))
        .build();
```

Caricamento di stream di dimensioni sconosciute

Il client asincrono S3 basato su Java con multipart abilitato è in grado di gestire in modo efficiente i flussi di input in cui la dimensione totale non è nota in anticipo:

```
public PutObjectResponse asyncClient_multipart_stream_unknown_size(String bucketName,
 String key, InputStream inputStream) {

    S3AsyncClient s3AsyncClient =
 S3AsyncClient.builder().multipartEnabled(true).build();
    ExecutorService executor = Executors.newSingleThreadExecutor();
    AsyncRequestBody body = AsyncRequestBody.fromInputStream(inputStream, null,
 executor); // 'null' indicates that the

    // content length is unknown.
    CompletableFuture<PutObjectResponse> responseFuture =
            s3AsyncClient.putObject(r -> r.bucket(bucketName).key(key), body)
                    .exceptionally(e -> {
                        if (e != null) {
                            logger.error(e.getMessage(), e);
                        }
                        return null;
                    });

    PutObjectResponse response = responseFuture.join(); // Wait for the response.
    executor.shutdown();
    return response;
}
```

Questo approccio previene i problemi che possono verificarsi quando si specifica manualmente una lunghezza errata del contenuto, come oggetti troncati o caricamenti non riusciti.

Trasferisci file e directory con Amazon S3 Transfer Manager

Amazon S3 Transfer Manager è un'utilità di trasferimento di 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 o sul client asincrono S3standard basato su Java con multipart abilitato, S3 Transfer Manager può trarre vantaggio dai miglioramenti delle prestazioni come l'API di caricamento multiparte e il recupero dell'intervallo 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 multiparte migliorate, configura il file di build con le dipendenze necessarie.

Use the AWS CRT-based S3 client

```
<dependencyManagement>
    <dependencies>
        <dependency>
            <groupId>software.amazon.awssdk</groupId>
            <artifactId>bom</artifactId>
            <version>2.27.21[1]</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>0.29.143[2]</version>
    </dependency>
</dependencies>
```

1 Ultima versione . 2 Ultima versione.

Use the Java-based S3 async client

```
<dependencyManagement>
    <dependencies>
        <dependency>
            <groupId>software.amazon.awssdk</groupId>
            <artifactId>bom</artifactId>
            <version>2.27.21[1]</version>
            <type>pom</type>
            <scope>import</scope>
        </dependency>
    </dependencies>
</dependencyManagement>
<dependencies>
    <dependency>
        <groupId>software.amazon.awssdk</groupId>
        <artifactId>s3-transfer-manager</artifactId>
    </dependency>
</dependencies>
```

[1] Ultima versione.

Crea un'istanza di S3 Transfer Manager

Per abilitare il trasferimento parallelo, è necessario passare un client S3 AWS basato su CRT OPPURE un client asincrono S3 basato su Java con multipart abilitato. Gli esempi seguenti mostrano come configurare un S3 Transfer Manager con impostazioni personalizzate.

Use the AWS CRT-based S3 client

```
        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();
```

Use the Java-based S3 async client

Se la `aws-crt` dipendenza non è inclusa nel file di build, S3 Transfer Manager si basa sul client asincrono S3 standard basato su Java utilizzato nell'SDK for Java 2.x.

Configurazione personalizzata del client S3: richiede l'attivazione di più parti

```
        S3AsyncClient s3AsyncClient = S3AsyncClient.builder()
                .multipartEnabled(true)
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_EAST_1)
                .build();

        S3TransferManager transferManager = S3TransferManager.builder()
                .s3Client(s3AsyncClient)
                .build();
```

Nessuna configurazione del client S3: supporto multiparte abilitato automaticamente

```
S3TransferManager transferManager = S3TransferManager.create();
```

Carica un file in un bucket S3

L'esempio seguente mostra un esempio di caricamento di file insieme all'uso opzionale di a 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'FileUploadoggetto restituito dal `uploadFile` metodo rappresenta il processo di caricamento. Al termine della richiesta, l'CompletedFileUploadoggetto 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 a LoggingTransferListener, che registra l'avanzamento del download.

Per scaricare un oggetto da un bucket S3 utilizzando S3 Transfer Manager, crea un DownloadFileRequestoggetto e passalo al metodo DownloadFile.

L'FileDownloadoggetto restituito dal `downloadFile` metodo `S3TransferManager`'s rappresenta il trasferimento del file. Una volta completato il download, CompletedFileDownloadcontiene l'accesso alle informazioni sul download.

```
    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().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 CopyRequestche possa essere utilizzato da S3 Transfer Manager.

L'`Copy`oggetto restituito dal `copy` metodo `S3TransferManager`'s rappresenta il processo di copia. Una volta completato il processo di copia, l'CompletedCopyoggetto 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'DirectoryUploadoggetto rappresenta il processo di caricamento, che genera un al CompletedDirectoryUploadtermine della richiesta. L'`CompleteDirectoryUpload`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'DirectoryDownloadoggetto rappresenta il processo di download, che genera un messaggio CompletedDirectoryDownloadquando 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.

Lavora con le notifiche degli eventi S3

Per aiutarti a monitorare l'attività nei tuoi bucket, Amazon S3 può inviare notifiche quando si verificano determinati eventi. La Guida per l'utente di Amazon S3 fornisce informazioni sulle notifiche che un bucket può inviare.

Puoi configurare un bucket per inviare eventi a quattro possibili destinazioni utilizzando l'SDK for Java:

- Argomenti su Amazon Simple Notification Service
- Code Amazon Simple Queue Service
- AWS Lambda funzioni
- Amazon EventBridge

Quando configuri un bucket a cui inviare eventi EventBridge, hai la possibilità di configurare una EventBridge regola per inviare lo stesso evento a più destinazioni. Quando configuri il bucket per l'invio diretto a una delle prime tre destinazioni, puoi specificare un solo tipo di destinazione per ogni evento.

Nella sezione successiva, vedrai come configurare un bucket utilizzando l'SDK for Java per inviare notifiche di eventi S3 in due modi: direttamente a una coda Amazon SQS e a. EventBridge

L'ultima sezione mostra come utilizzare l'API S3 Event Notifications per utilizzare le notifiche in modo orientato agli oggetti.

Configura un bucket da inviare direttamente a una destinazione

L'esempio seguente configura un bucket per inviare notifiche quando si verificano eventi di creazione di oggetti o eventi di tagging di oggetti su un bucket.

```
static void processS3Events(String bucketName, String queueArn) {
    // Configure the bucket to send Object Created and Object Tagging notifications to
 an existing SQS queue.
    s3Client.putBucketNotificationConfiguration(b -> b
            .notificationConfiguration(ncb -> ncb
                    .queueConfigurations(qcb -> qcb
                            .events(Event.S3_OBJECT_CREATED, Event.S3_OBJECT_TAGGING)
                            .queueArn(queueArn)))
                    .bucket(bucketName)
    );
}
```

Il codice mostrato sopra imposta una coda per ricevere due tipi di eventi. Comodamente, il `queueConfigurations` metodo consente di impostare più destinazioni di coda, se necessario. Inoltre, nel `notificationConfiguration` metodo puoi impostare destinazioni aggiuntive, come uno o più argomenti di Amazon SNS o una o più funzioni Lambda. Il seguente frammento mostra un esempio con due code e tre tipi di destinazioni.

```
s3Client.putBucketNotificationConfiguration(b -> b
                .notificationConfiguration(ncb -> ncb
                        .queueConfigurations(qcb -> qcb
                                .events(Event.S3_OBJECT_CREATED,
 Event.S3_OBJECT_TAGGING)
                                .queueArn(queueArn),
                                qcb2 -> qcb2.<...>)
                        .topicConfigurations(tcb -> tcb.<...>)
                        .lambdaFunctionConfigurations(lfcb -> lfcb.<...>))
                        .bucket(bucketName)
        );
```

Il GitHub repository Code Examples contiene l'esempio completo di invio di notifiche di eventi S3 direttamente a una coda.

Configura un bucket a cui inviare EventBridge

L'esempio seguente configura un bucket a cui inviare notifiche. EventBridge

```
public static String setBucketNotificationToEventBridge(String bucketName) {
    // Enable bucket to emit S3 Event notifications to EventBridge.
    s3Client.putBucketNotificationConfiguration(b -> b
            .bucket(bucketName)
            .notificationConfiguration(b1 -> b1
                    .eventBridgeConfiguration(SdkBuilder::build))
    .build());
```

Quando si configura un bucket a cui inviare eventi EventBridge, è sufficiente indicare la EventBridge destinazione, non i tipi di eventi né la destinazione finale a cui EventBridge verranno inviati gli eventi. Puoi configurare gli obiettivi finali e i tipi di eventi utilizzando il client di Java SDK. EventBridge

Il codice seguente mostra come EventBridge configurare la suddivisione degli eventi creati dagli oggetti in un argomento e in una coda.

```
   public static String configureEventBridge(String topicArn, String queueArn) {
        try {
            // Create an EventBridge rule to route Object Created notifications.
            PutRuleRequest putRuleRequest = PutRuleRequest.builder()
                    .name(RULE_NAME)
                    .eventPattern("""
                            {
                              "source": ["aws.s3"],
                              "detail-type": ["Object Created"],
                              "detail": {
                                "bucket": {
                                  "name": ["%s"]
                                }
                              }
                            }
                            """.formatted(bucketName))
                    .build();

            // Add the rule to the default event bus.
            PutRuleResponse putRuleResponse = eventBridgeClient.putRule(putRuleRequest)
                    .whenComplete((r, t) -> {
                        if (t != null) {
                            logger.error("Error creating event bus rule: " +
 t.getMessage(), t);
                            throw new RuntimeException(t.getCause().getMessage(), t);
                        }
```

```
                        logger.info("Event bus rule creation request sent successfully.
 ARN is: {}", r.ruleArn());
                    }).join();

            // Add the existing SNS topic and SQS queue as targets to the rule.
            eventBridgeClient.putTargets(b -> b
                    .eventBusName("default")
                    .rule(RULE_NAME)
                    .targets(List.of (
                            Target.builder()
                                    .arn(queueArn)
                                    .id("Queue")
                                    .build(),
                            Target.builder()
                                    .arn(topicArn)
                                    .id("Topic")
                                    .build())
                            )
                    ).join();
            return putRuleResponse.ruleArn();
        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return null;
    }
```

Per utilizzarlo EventBridge nel codice Java, aggiungi una dipendenza dall'`eventbridge`artefatto al tuo file Maven. `pom.xml`

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

Il GitHub repository Code Examples contiene l'esempio completo di invio di notifiche di eventi S3 e quindi a un argomento EventBridge e a una coda.

Utilizza l'API S3 Event Notifications per elaborare gli eventi

Dopo che una destinazione riceve eventi di notifica S3, puoi elaborarli in modo orientato agli oggetti utilizzando l'API S3 Event Notifications. Puoi utilizzare l'API S3 Event Notifications per lavorare con le

notifiche di eventi che vengono inviate direttamente a un target (come mostrato nel primo esempio), ma non con le notifiche inoltrate. EventBridge Le notifiche di eventi S3 inviate dai bucket EventBridge contengono una struttura diversa che l'API S3 Event Notifications attualmente non gestisce.

Aggiungi dipendenza

L'API S3 Event Notifications è stata rilasciata con la versione 2.25.11 dell'SDK for Java 2.x.

Per utilizzare l'API S3 Event Notifications, aggiungi l'elemento di dipendenza richiesto a `pom.xml` Maven, come mostrato nel seguente frammento.

```
<dependencyManagement>
    <dependencies>
        <dependency>
            <groupId>software.amazon.awssdk</groupId>
            <artifactId>bom</artifactId>
            <version>2.X.X[1]</version>
            <type>pom</type>
            <scope>import</scope>
        </dependency>
    </dependencies>
</dependencyManagement>
<dependencies>
    <dependency>
        <groupId>software.amazon.awssdk</groupId>
        <artifactId>s3-event-notifications</artifactId>
    </dependency>
</dependencies>
```

1 Ultima versione.

Usa la **`S3EventNotification`** classe

Crea un'**`S3EventNotification`**istanza da una stringa JSON

Per convertire una stringa JSON in un `S3EventNotification` oggetto, utilizzate i metodi statici della `S3EventNotification` classe come mostrato nell'esempio seguente.

```
import software.amazon.awssdk.eventnotifications.s3.model.S3EventNotification
import software.amazon.awssdk.eventnotifications.s3.model.S3EventNotificationRecord
import software.amazon.awssdk.services.sqs.model.Message;
```

```
public class S3EventNotificationExample {
    ...

    void receiveMessage(Message message) {
       // Message received from SQSClient.
       String sqsEventBody = message.body();
       S3EventNotification s3EventNotification =
 S3EventNotification.fromJson(sqsEventBody);

       // Use getRecords() to access all the records in the notification.


       List<S3EventNotificationRecord> records = s3EventNotification.getRecords();

        S3EventNotificationRecord record = records.stream().findFirst();
        // Use getters on the record to access individual attributes.
        String awsRegion = record.getAwsRegion();
        String eventName = record.getEventName();
        String eventSource = record.getEventSource();

    }
}
```

In questo esempio, il `fromJson` metodo converte la stringa JSON in un oggetto. `S3EventNotification` I campi mancanti nella stringa JSON produrranno `null` valori nei campi degli oggetti Java corrispondenti e tutti i campi aggiuntivi nel JSON verranno ignorati.

Ulteriori informazioni APIs relative al record di notifica degli eventi sono disponibili nel riferimento API per. [S3EventNotificationRecord]

Convertire un'**`S3EventNotification`**istanza in una stringa JSON

Utilizzate il metodo `toJson` (or`toJsonPretty`) per convertire un `S3EventNotification` oggetto in una stringa JSON, come mostrato nell'esempio seguente.

```
import software.amazon.awssdk.eventnotifications.s3.model.S3EventNotification

public class S3EventNotificationExample {
    ...

    void toJsonString(S3EventNotification event) {
```

```
        String json = event.toJson();
        String jsonPretty = event.toJsonPretty();

        System.out.println("JSON: " + json);
        System.out.println("Pretty JSON: " + jsonPretty);
    }
}
```

I campi per `GlacierEventDataReplicationEventData`,`IntelligentTieringEventData`, e `LifecycleEventData` sono esclusi dal JSON se lo sono. `null` Gli altri `null` campi verranno serializzati come. `null`

Di seguito viene mostrato un esempio di output del `toJsonPretty` metodo per un evento di tagging di oggetti S3.

```
{
  "Records" : [ {
    "eventVersion" : "2.3",
    "eventSource" : "aws:s3",
    "awsRegion" : "us-east-1",
    "eventTime" : "2024-07-19T20:09:18.551Z",
    "eventName" : "ObjectTagging:Put",
    "userIdentity" : {
      "principalId" : "AWS:XXXXXXXXXXX"
    },
    "requestParameters" : {
      "sourceIPAddress" : "XXX.XX.XX.XX"
    },
    "responseElements" : {
      "x-amz-request-id" : "XXXXXXXXXXXX",
      "x-amz-id-2" : "XXXXXXXXXXXXX"
    },
    "s3" : {
      "s3SchemaVersion" : "1.0",
      "configurationId" : "XXXXXXXXXXXX",
      "bucket" : {
        "name" : "DOC-EXAMPLE-BUCKET",
        "ownerIdentity" : {
          "principalId" : "XXXXXXXXXX"
        },
        "arn" : "arn:aws:s3:::XXXXXXXXX"
      },
      "object" : {
```

```
        "key" : "akey",
        "size" : null,
        "eTag" : "XXXXXXXXXX",
        "versionId" : null,
        "sequencer" : null
      }
    }
  } ]
}
```

È disponibile un esempio completo GitHub che mostra come utilizzare l'API per gestire le notifiche ricevute da una coda Amazon SQS.

Elabora gli eventi S3 in Lambda con le librerie Java: e AWS SDK for Java 2.x`aws-lambda-java-events`

Invece di utilizzare l'SDK for Java 2.x per elaborare le notifiche degli eventi di Amazon S3 in una funzione Lambda, puoi utilizzare la libreria nella versione 3.x.x. `aws-lambda-java-events` AWS gestisce la `aws-lambda-java-events` libreria in modo indipendente e ha i propri requisiti di dipendenza. La `aws-lambda-java-events` libreria funziona solo con gli eventi S3 nelle funzioni Lambda, mentre l'SDK per Java 2.x funziona con gli eventi S3 nelle funzioni Lambda, Amazon SNS e Amazon SQS.

Entrambi gli approcci modellano il payload di notifica degli eventi JSON in modo orientato agli oggetti con similari. APIs La tabella seguente mostra le notevoli differenze tra l'utilizzo dei due approcci.

	AWS SDK per Java	aws-lambda-java-events libreria
Denominazione dei pacchetti	`software.amazon.awssdk.eventnotifications.s3.model.S3EventNotification`	`com.amazonaws.services.lambda.runtime.events.models.s3.S3EventNotification`
Parametro `RequestHandler`	Scrivi l'`RequestHandler` implementazione della tua funzione Lambda per ricevere una stringa JSON:	Scrivi l'`RequestHandler` implementazione della tua funzione Lambda per ricevere un `S3Event` oggetto:

	AWS SDK per Java	aws-lambda-java-events libreria

	`import com.amazonaws.services.lambda.runtime.Context;`
	`import com.amazonaws.services.lambda.runtime.RequestHandler;`
	`import software.amazon.awssdk.eventnotifications.s3.model.S3EventNotification;`
	
	`public class Handler implements RequestHandler<String, String> {`
	
	`@Override`
	`public String handleRequest(String jsonS3Event, Context context) {`
	`S3EventNotification s3Event = S3EventNotification`
	
	`.fromJson(jsonS3Event);`
	`// Work with the s3Event object.`
	
	`...`
	`}`
	`}`
		`import com.amazonaws.services.lambda.runtime.Context;`
		`import com.amazonaws.services.lambda.runtime.RequestHandler;`
		`import com.amazonaws.services.lambda.runtime.events.S3Event;`
		
		`public class Handler implements RequestHandler<S3Event, String> {`
		
		`@Override`
		`public String handleRequest(S3Event s3event, Context context) {`
		`// Work with the s3Event object.`
		
		`...`
		`}`
		`}`

	AWS SDK per Java	aws-lambda-java-events libreria
Dipendenze da Maven	<dependencyManagement> <dependencies> <dependency> <groupId>software.amazon.awssdk</groupId> <artifactId>bom</artifactId> <version>2.X.X</version> <type>pom</type> <scope>import</scope> </dependency> </dependencies> </dependencyManagement> <dependencies> <dependency> <groupId>software.amazon.awssdk</groupId> <artifactId>s3-event-notifications</artifactId> </dependency> <!-- Add other SDK dependencies that you need. --> </dependencies>	<dependencyManagement> <dependencies> <dependency> <groupId>software.amazon.awssdk</groupId> <artifactId>bom</artifactId> <version>2.X.X</version> <type>pom</type> <scope>import</scope> </dependency> </dependencies> </dependencyManagement> <dependencies> <!-- The following two dependencies are for the aws-lambda-java-events library. --> <dependency> <groupId>com.amazonaws</groupId> <artifactId>aws-lambda-java-core</artifactId> <version>1.2.3</version> </dependency> <dependency> <groupId>com.amazonaws</groupId>

	AWS SDK per Java	aws-lambda-java-events libreria
		<artifact Id>aws-lambda-java-events</artifactId> <version> 3.15.0</version> </dependency> <!-- Add other SDK dependencies that you need. --> </dependencies>

Lavora con Amazon Simple Notification Service

Con Amazon Simple Notification Service, puoi inviare facilmente messaggi di notifica in tempo reale dalle tue 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 a quali sistemi inviare un messaggio, ad esempio suddividendo un messaggio AWS Lambda e un webhook HTTP. I messaggi vengono inviati ai Amazon SNS canali definiti nell'argomento e poi vengono distribuiti ai canali definiti nell'argomento. Questo rende i messaggi disponibili per gli abbonati.

Per creare un argomento, crea prima un CreateTopicRequestoggetto, con il nome dell'argomento impostato utilizzando il `name()` metodo nel generatore. Quindi, invia l'oggetto della richiesta a Amazon SNS utilizzando il `createTopic()` metodo di. SnsClient È possibile acquisire il risultato di questa richiesta come CreateTopicResponseoggetto, come illustrato 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 ListTopicsRequestoggetto. Quindi, invia l'oggetto della richiesta a Amazon SNS utilizzando il `listTopics()` metodo di. `SnsClient` È possibile acquisire il risultato di questa richiesta come ListTopicsResponseoggetto.

Il seguente frammento di codice stampa il codice di stato HTTP della richiesta e un elenco di Amazon Resource Names (ARNs) per i tuoi Amazon SNS argomenti.

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);
        }
    }
```

Guarda 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 averli Amazon SNS ricevuti.

Per configurare un canale di comunicazione come endpoint per un argomento, sottoscrivi tale endpoint all'argomento. Per iniziare, costruisci un SubscribeRequestoggetto. Specificate il canale di comunicazione (ad esempio, `lambda` o`email`) 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()` Inviate l'oggetto della richiesta a Amazon SNS utilizzando il `subscribe()` metodo di. `SnsClient` È possibile acquisire il risultato di questa richiesta come SubscribeResponseoggetto.

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 PublishRequestoggetto. Specifica `message()` per l'invio e l'ARN dell'argomento (`topicArn()`) a cui inviarlo. Quindi, invia l'oggetto della richiesta a Amazon SNS utilizzando il `publish()` metodo di`SnsClient`. È possibile acquisire il risultato di questa richiesta come PublishResponseoggetto.

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 UnsubscribeRequestoggetto 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 DeleteTopicRequestoggetto con l'ARN dell'argomento impostato come `topicArn()` metodo nel generatore. Quindi invia l'oggetto della richiesta a Amazon SNS utilizzando il `deleteTopic()` metodo di. `SnsClient` È possibile acquisire il risultato di questa richiesta come DeleteTopicResponseoggetto, come illustrato 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 Servicecon AWS SDK per 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

- Utilizza il batching automatico delle richieste per Amazon SQS con AWS SDK for Java 2.x
- Lavora con le code di Amazon Simple Queue Service messaggi
- Inviare, ricevere ed eliminare Amazon Simple Queue Service messaggi

Utilizza il batching automatico delle richieste per Amazon SQS con AWS SDK for Java 2.x

L'API Automatic Request Batching per Amazon SQS è una libreria di alto livello che fornisce un modo efficiente per creare batch e bufferizzare le richieste per le operazioni SQS. Utilizzando l'API di batching, riduci il numero di richieste a SQS, il che migliora la produttività e minimizza i costi.

Poiché i metodi dell'API batch corrispondono ai `SqsAsyncClient` metodi—`sendMessage`,, `changeMessageVisibility``deleteMessage`, `receiveMessage` —puoi utilizzare l'API batch come sostituto immediato con modifiche minime.

Questo argomento offre una panoramica su come configurare e utilizzare l'API Automatic Request Batching per Amazon SQS.

Verifica i prerequisiti

È necessario utilizzare la versione 2.28.0 o successiva dell'SDK for Java 2.x per avere accesso all'API di batch. Il tuo Maven `pom.xml` dovrebbe contenere almeno i seguenti elementi.

```
<dependencyManagement>
    <dependencies>
        <dependency>
            <groupId>software.amazon.awssdk</groupId>
            <artifactId>bom</artifactId>
            <version>2.28.23[1]</version>
            <type>pom</type>
            <scope>import</scope>
        </dependency>
    </dependencies>
</dependencyManagement>
<dependencies>
    <dependency>
        <groupId>software.amazon.awssdk</groupId>
        <artifactId>sqs</artifactId>
    </dependency>
</dependencies>
```

1 Ultima versione

Crea un gestore di batch

L'API di raggruppamento automatico delle richieste è implementata dall'SqsAsyncBatchManagerinterfaccia. È possibile creare un'istanza del gestore in due modi.

Configurazione predefinita utilizzando **`SqsAsyncClient`**

Il modo più semplice per creare un gestore di batch consiste nel chiamare il metodo `batchManager` factory su un'SqsAsyncClientistanza esistente. L'approccio semplice è illustrato nel frammento seguente.

```
SqsAsyncClient asyncClient = SqsAsyncClient.create();
SqsAsyncBatchManager sqsAsyncBatchManager = asyncClient.batchManager();
```

Quando utilizzate questo approccio, l'`SqsAsyncBatchManager`istanza utilizza i valori predefiniti mostrati nella tabella della sezione. the section called “Impostazioni di configurazione” Inoltre, l'`SqsAsyncBatchManager`istanza utilizza `ExecutorService` l'`SqsAsyncClient`istanza da cui è stata creata.

Configurazione personalizzata utilizzando **`SqsAsyncBatchManager.Builder`**

Per casi d'uso più avanzati, è possibile personalizzare il gestore batch utilizzando `SqsAsyncBatchManager.Builder`. Utilizzando questo approccio per creare un'`SqsAsyncBatchManager`istanza, è possibile ottimizzare il comportamento di batch. Il seguente frammento mostra un esempio di come utilizzare il builder per personalizzare il comportamento del batch.

```
SqsAsyncBatchManager batchManager = SqsAsyncBatchManager.builder()
    .client(SqsAsyncClient.create())
    .scheduledExecutor(Executors.newScheduledThreadPool(5))
    .overrideConfiguration(b -> b
        .receiveMessageMinWaitDuration(Duration.ofSeconds(10))
        .receiveMessageVisibilityTimeout(Duration.ofSeconds(1))
        .receiveMessageAttributeNames(Collections.singletonList("*"))

 .receiveMessageSystemAttributeNames(Collections.singletonList(MessageSystemAttributeName.ALL))
    .build();
```

Quando utilizzate questo approccio, potete regolare le impostazioni sull'`BatchOverrideConfiguration`oggetto mostrate nella tabella della sezione. the section called “Impostazioni di configurazione” È inoltre possibile fornire una personalizzazione `ScheduledExecutorService`per il gestore dei batch utilizzando questo approccio.

Inviare messaggi

Per inviare messaggi con il gestore batch, usa il `SqsAsyncBatchManager#sendMessage` metodo. L'SDK memorizza le richieste nel buffer e le invia come batch quando vengono raggiunti `sendRequestFrequency` i valori `maxBatchSize` or.

L'esempio seguente mostra una `sendMessage` richiesta immediatamente successiva a un'altra richiesta. In questo caso, l'SDK invia entrambi i messaggi in un unico batch.

```
// Sending the first message
CompletableFuture<SendMessageResponse> futureOne =
    sqsAsyncBatchManager.sendMessage(r -> r.messageBody("One").queueUrl("queue"));

// Sending the second message
CompletableFuture<SendMessageResponse> futureTwo =
    sqsAsyncBatchManager.sendMessage(r -> r.messageBody("Two").queueUrl("queue"));

// Waiting for both futures to complete and retrieving the responses
SendMessageResponse messageOne = futureOne.join();
SendMessageResponse messageTwo = futureTwo.join();
```

Modifica il timeout di visibilità dei messaggi

È possibile modificare il timeout di visibilità dei messaggi in un batch utilizzando il SqsAsyncBatchManager#changeMessageVisibilitymetodo. L'SDK memorizza le richieste nel buffer e le invia in batch quando vengono raggiunti `sendRequestFrequency` i valori `maxBatchSize` or.

L'esempio seguente mostra come chiamare il `changeMessageVisibility` metodo.

```
CompletableFuture<ChangeMessageVisibilityResponse> futureOne =
    sqsAsyncBatchManager.changeMessageVisibility(r ->
        r.receiptHandle("receiptHandle")
         .queueUrl("queue"));
ChangeMessageVisibilityResponse response = futureOne.join();
```

Eliminare i messaggi

È possibile eliminare i messaggi in batch utilizzando il SqsAsyncBatchManager#deleteMessagemetodo. L'SDK memorizza le richieste nel buffer e le invia come batch quando vengono raggiunti `sendRequestFrequency` i valori `maxBatchSize` or.

L'esempio seguente mostra come chiamare il `deleteMessage` metodo.

```
CompletableFuture<DeleteMessageResponse> futureOne =
    sqsAsyncBatchManager.deleteMessage(r ->
        r.receiptHandle("receiptHandle")
         .queueUrl("queue"));
DeleteMessageResponse response = futureOne.join();
```

Ricevere messaggi

Usa le impostazioni predefinite

Quando esegui il polling del `SqsAsyncBatchManager#receiveMessage` metodo nell'applicazione, il gestore batch recupera i messaggi dal suo buffer interno, che l'SDK aggiorna automaticamente in background.

L'esempio seguente mostra come chiamare il metodo. `receiveMessage`

```
CompletableFuture<ReceiveMessageResponse> responseFuture =
    sqsAsyncBatchManager.receiveMessage(r -> r.queueUrl("queueUrl"));
```

Usa impostazioni personalizzate

Se desideri personalizzare ulteriormente la richiesta, ad esempio impostando tempi di attesa personalizzati e specificando il numero di messaggi da recuperare, puoi personalizzare la richiesta come mostrato nell'esempio seguente.

```
CompletableFuture<ReceiveMessageResponse> response =
    sqsAsyncBatchManager.receiveMessage(r ->
        r.queueUrl("queueUrl")
         .waitTimeSeconds(5)
         .visibilityTimeout(20));
```

> **Note**
>
> Se chiami `receiveMessage` con un `ReceiveMessageRequest` che include uno dei seguenti parametri, l'SDK ignora il gestore batch e invia una normale richiesta asincrona: `receiveMessage`
>
> - `messageAttributeNames`
> - `messageSystemAttributeNames`
> - `messageSystemAttributeNamesWithStrings`
> - `overrideConfiguration`

Sostituisci le impostazioni di configurazione per `SqsAsyncBatchManager`

È possibile modificare le seguenti impostazioni quando si crea un'`SqsAsyncBatchManager`istanza. Il seguente elenco di impostazioni è disponibile in `BatchOverrideConfiguration.Builder`.

Impostazione	Descrizione	Valore predefinito
`maxBatchSize`	Numero massimo di richieste per batch per ciascuna `SendMessageBatchRequest` `ChangeMessageVisibilityBatchRequest` , o`DeleteMessageBatchRequest` . Il valore massimo è 10.	10
`sendRequestFrequency`	Tempo prima dell'invio di un batch, a meno che non `maxBatchSize` venga raggiunto prima. Valori più alti possono ridurre le richieste ma aumentare la latenza.	200 ms
`receiveMessageVisibilityTimeout`	Timeout di visibilità per i messaggi. Se non è impostato , viene utilizzato il valore predefinito della coda.	L'impostazione predefinita della coda
`receiveMessageMinWaitDuration`	Tempo di attesa minimo per le `receiveMessage` richieste . Evita di impostarlo su 0 per evitare sprechi di CPU.	50 ms
`receiveMessageSystemAttributeNames`	Elenco dei nomi degli attributi di sistema da richiedere per `receiveMessage` le chiamate.	Nessuno

Impostazione	Descrizione	Valore predefinito
`receiveMessageAttributeNames`	Elenco dei nomi degli attributi da richiedere per `receiveMessage` le chiamate.	Nessuno

Lavora con le code di Amazon Simple Queue Service messaggi

Una coda di messaggi è il contenitore logico utilizzato per inviare messaggi in modo affidabile. 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 Service sviluppatori.

Questo argomento descrive come creare, elencare, eliminare e ottenere l'URL di una Amazon Simple Queue Service coda utilizzando. AWS SDK per 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

- CreateQueuenell' Amazon Simple Queue Service API Reference
- GetQueueUrlnell' Amazon Simple Queue Service API Reference
- ListQueuesnell' Amazon Simple Queue Service API Reference
- DeleteQueuenell' 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 SendMessageRequestoggetto 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 SendMessageBatchRequestche 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 ReceiveMessageRequestche 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)
                .maxNumberOfMessages(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, eliminatelo 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
- SendMessagenell' Amazon Simple Queue Service API Reference
- SendMessageBatchnell' Amazon Simple Queue Service API Reference
- ReceiveMessagenell' Amazon Simple Queue Service API Reference
- DeleteMessagenell' Amazon Simple Queue Service API Reference

Lavora con 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 Amazon Transcribe in streaming 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 editore che pubblica dati audio dal flusso Amazon Transcribe audio.

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();
        }
    }
}
```

Vedi 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 StartStreamTranscriptionResponseHandlerfile 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.StartStreamTranscriptionResponseHand
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 nozioni di base sono esempi di codice che mostrano come eseguire le operazioni essenziali all'interno di un servizio.

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.

Gli scenari sono esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio o combinate con altri Servizi AWS.

Alcuni servizi contengono ulteriori categorie di esempi che mostrano come sfruttare le librerie o le funzioni specifiche del servizio.

Servizi

- Esempi ACM che utilizzano SDK for Java 2.x
- 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
- AWS Batch esempi che utilizzano 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 Firehose che utilizzano SDK for Java 2.x
- Esempi di Amazon DocumentDB con SDK for Java 2.x
- Esempi di DynamoDB con SDK for Java 2.x
- EC2 Esempi di Amazon che utilizzano SDK for Java 2.x
- Esempi di Amazon ECR con SDK for Java 2.x
- Esempi di Amazon ECS che utilizzano 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
- AWS Entity Resolution 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
- EventBridge Esempi di scheduler 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
- AWS IoT FleetWise esempi che utilizzano SDK for Java 2.x
- AWS IoT SiteWise 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
- Esempi di Amazon Lex con SDK for Java 2.x
- Esempi di Amazon Location con SDK for Java 2.x
- Location Service Places: esempi con SDK for Java 2.x
- Marketplace AWS Esempi di API di catalogo che utilizzano SDK for Java 2.x
- Marketplace AWS Esempi di API Agreement che utilizzano 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 MSK con SDK for Java 2.x
- Esempi di Neptune con SDK for Java 2.x
- Esempi di Partner Central 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 RDS Data Service con 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 Amazon S3 Control con SDK for Java 2.x
- Esempi di S3 Directory Bucket che utilizzano SDK for Java 2.x
- Esempi di S3 Glacier con SDK for Java 2.x
- SageMaker Esempi di intelligenza artificiale 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
- Supporto 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 Amazon Transcribe Streaming con SDK for Java 2.x
- Esempi di Amazon Translate con SDK for Java 2.x

Esempi ACM 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 ACM.

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.

Ogni esempio include un collegamento al codice sorgente completo, in cui è possibile trovare istruzioni su come configurare ed eseguire il codice nel contesto.

Argomenti

- Azioni

Azioni

`AddTagsToCertificate`

Il seguente esempio di codice mostra come utilizzare`AddTagsToCertificate`.

SDK per Java 2.x

> **Note**
>
> C'è 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.
 * <p>
```

```
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class AddTagsToCertificate {

    public static void main(String[] args) {
        final String usage = """

            Usage:    <certArn>

            Where:
                certArn - the ARN of the certificate.
            """;
        if (args.length != 1) {
            System.out.println(usage);
            return;
        }

        String certArn = args[0];
        addTags(certArn);
    }

    /**
     * Adds tags to a certificate in AWS Certificate Manager (ACM).
     *
     * @param certArn the Amazon Resource Name (ARN) of the certificate to add tags
 to
     */
    public static void addTags(String certArn) {
        AcmClient acmClient = AcmClient.create();
        List<Tag> expectedTags =
 List.of(Tag.builder().key("key").value("value").build());
        AddTagsToCertificateRequest addTagsToCertificateRequest =
 AddTagsToCertificateRequest.builder()
            .certificateArn(certArn)
            .tags(expectedTags)
            .build();

        try {
            acmClient.addTagsToCertificate(addTagsToCertificateRequest);
            System.out.println("Successfully added tags to a certificate");
        } catch (AcmException e) {
            System.out.println(e.getMessage());
```

```
        }
    }
}
```

- Per i dettagli sull'API, consulta la AddTagsToCertificatesezione 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.

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteCert {

    public static void main(String[] args) {
        final String usage = """

            Usage:    <certArn>

            Where:
                certArn - the ARN of the certificate.
            """;
        if (args.length != 1) {
            System.out.println(usage);
            return;
```

```
        }

        String certArn = args[0];
        deleteCertificate(certArn);
    }

    /**
     * Deletes an SSL/TLS certificate from the AWS Certificate Manager (ACM).
     *
     * @param certArn the Amazon Resource Name (ARN) of the certificate to be
 deleted
     */
    public static void deleteCertificate( String certArn) {
        AcmClient acmClient = AcmClient.create();
        DeleteCertificateRequest request = DeleteCertificateRequest.builder()
            .certificateArn(certArn)
            .build();

        try {
            acmClient.deleteCertificate(request);
            System.out.println("The certificate was deleted");

        } catch (AcmException e) {
            System.out.println(e.getMessage());
        }
    }
}
```

- Per i dettagli sull'API, consulta la DeleteCertificatesezione AWS SDK for Java 2.x API Reference.

`DescribeCertificate`

Il seguente esempio di codice mostra come utilizzare`DescribeCertificate`.

SDK per Java 2.x

> **Note**
>
> C'è 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.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class DescribeCert {

    public static void main(String[] args) {
        final String usage = """

            Usage:    <certArn>

            Where:
                certArn - the ARN of the certificate.
            """;
        if (args.length != 1) {
            System.out.println(usage);
            return;
        }

        String certArn = args[0];
        describeCertificate(certArn);
    }

    /**
     * Describes the details of an SSL/TLS certificate.
     *
     * @param certArn the Amazon Resource Name (ARN) of the certificate to describe
     * @throws AcmException if an error occurs while describing the certificate
```

```
     */
    public static void describeCertificate(String certArn) {
        AcmClient acmClient = AcmClient.create();
        DescribeCertificateRequest req = DescribeCertificateRequest.builder()
            .certificateArn(certArn)
            .build();

        try {
            DescribeCertificateResponse response =
 acmClient.describeCertificate(req);

            // Print the certificate details.
            System.out.println("Certificate ARN: " +
 response.certificate().certificateArn());
            System.out.println("Domain Name: " +
 response.certificate().domainName());
            System.out.println("Issued By: " + response.certificate().issuer());
            System.out.println("Issued On: " + response.certificate().issuedAt());
            System.out.println("Status: " + response.certificate().status());
        } catch (AcmException e) {
            System.out.println(e.getMessage());
        }
    }
}
```

- Per i dettagli sull'API, consulta la DescribeCertificatesezione AWS SDK for Java 2.x API Reference.

`ExportCertificate`

Il seguente esempio di codice mostra come utilizzare`ExportCertificate`.

SDK per Java 2.x

> **Note**
>
> C'è 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.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ExportCertificate {

    public static void main(String[] args) throws Exception {
        final String usage = """

            Usage:    <certArn>

            Where:
                certArn - the ARN of the certificate.
            """;
        if (args.length != 1) {
            System.out.println(usage);
            return;
        }

        String certArn = args[0];
        exportCert(certArn);
    }

    /**
     * Exports an SSL/TLS certificate and its associated private key and certificate
 chain from AWS Certificate Manager (ACM).
     *
     * @param certArn The Amazon Resource Name (ARN) of the certificate that you
 want to export.
     * @throws IOException If an I/O error occurs while reading the private key
 passphrase file or exporting the certificate.
     */
    public static void exportCert(String certArn) throws IOException {
        AcmClient acmClient = AcmClient.create();

        // Initialize a file descriptor for the passphrase file.
        RandomAccessFile filePassphrase = null;
        ByteBuffer bufPassphrase = null;
```

```
        // Create a file stream for reading the private key passphrase.
        try {
            filePassphrase = new RandomAccessFile("C:\\AWS\\password.txt", "r");
        } catch (IllegalArgumentException | SecurityException |
FileNotFoundException ex) {
            throw ex;
        }

        // Create a channel to map the file.
        FileChannel channelPassphrase = filePassphrase.getChannel();

        // Map the file to the buffer.
        try {
            bufPassphrase = channelPassphrase.map(FileChannel.MapMode.READ_ONLY, 0,
channelPassphrase.size());
            channelPassphrase.close();
            filePassphrase.close();
        } catch (IOException ex) {
            throw ex;
        }

        // Create a request object.
        ExportCertificateRequest req = ExportCertificateRequest.builder()
            .certificateArn(certArn)
            .passphrase(SdkBytes.fromByteBuffer(bufPassphrase))
            .build();

        // Export the certificate.
        ExportCertificateResponse result = null;
        try {
            result = acmClient.exportCertificate(req);
        } catch (InvalidArnException | InvalidTagException |
ResourceNotFoundException ex) {
            throw ex;
        }

        // Clear the buffer.
        bufPassphrase.clear();

        // Display the certificate and certificate chain.
        String certificate = result.certificate();
        System.out.println(certificate);
```

```
        String certificateChain = result.certificateChain();
        System.out.println(certificateChain);

        // This example retrieves but does not display the private key.
        String privateKey = result.privateKey();
        System.out.println("The example is complete");
    }
}
```

- Per i dettagli sull'API, consulta la ExportCertificatesezione AWS SDK for Java 2.x API Reference.

`ImportCertificate`

Il seguente esempio di codice mostra come utilizzare`ImportCertificate`.

SDK per Java 2.x

> **Note**
>
> C'è 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.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ImportCert {

    public static void main(String[] args) {
        final String usage = """
            Usage: <bucketName> <certificateKey> <privateKeyKey>

            Where:
```

```
                bucketName - The name of the S3 bucket containing the certificate
and private key.
                certificateKey - The object key for the SSL/TLS certificate file in
S3.
                privateKeyKey - The object key for the private key file in S3.
            """;

        //if (args.length != 3) {
        //    System.out.println(usage);
        //    return;
       // }

        String bucketName = "certbucket100" ; //args[0];
        String certificateKey = "certificate.pem" ; // args[1];
        String privateKeyKey = "private_key.pem" ; //args[2];

        String certificateArn = importCertificate(bucketName, certificateKey,
privateKeyKey);
        System.out.println("Certificate imported with ARN: " + certificateArn);
    }

    /**
     * Imports an SSL/TLS certificate and private key from S3 into AWS Certificate
Manager (ACM).
     *
     * @param bucketName     The name of the S3 bucket.
     * @param certificateKey The key for the SSL/TLS certificate file in S3.
     * @param privateKeyKey  The key for the private key file in S3.
     * @return The ARN of the imported certificate.
     */
    public static String importCertificate(String bucketName, String certificateKey,
String privateKeyKey) {
        AcmClient acmClient = AcmClient.create();
        S3Client s3Client = S3Client.create();

        try {
            byte[] certificateBytes = downloadFileFromS3(s3Client, bucketName,
certificateKey);
            byte[] privateKeyBytes = downloadFileFromS3(s3Client, bucketName,
privateKeyKey);

            ImportCertificateRequest request = ImportCertificateRequest.builder()

.certificate(SdkBytes.fromByteBuffer(ByteBuffer.wrap(certificateBytes)))
```

```
 .privateKey(SdkBytes.fromByteBuffer(ByteBuffer.wrap(privateKeyBytes)))
                    .build();

            ImportCertificateResponse response =
 acmClient.importCertificate(request);
            return response.certificateArn();

        } catch (IOException e) {
            System.err.println("Error downloading certificate or private key from
 S3: " + e.getMessage());
        } catch (S3Exception e) {
            System.err.println("S3 error: " + e.awsErrorDetails().errorMessage());
        }
        return "";
    }

    /**
     * Downloads a file from Amazon S3 and returns its contents as a byte array.
     *
     * @param s3Client   The S3 client.
     * @param bucketName The name of the S3 bucket.
     * @param objectKey  The key of the object in S3.
     * @return The file contents as a byte array.
     * @throws IOException If an I/O error occurs.
     */
    private static byte[] downloadFileFromS3(S3Client s3Client, String bucketName,
 String objectKey) throws IOException {
        GetObjectRequest getObjectRequest = GetObjectRequest.builder()
                .bucket(bucketName)
                .key(objectKey)
                .build();

        try (ResponseInputStream<GetObjectResponse> s3Object =
 s3Client.getObject(getObjectRequest);
             ByteArrayOutputStream byteArrayOutputStream = new
 ByteArrayOutputStream()) {
            IoUtils.copy(s3Object, byteArrayOutputStream);
            return byteArrayOutputStream.toByteArray();
        }
    }
}
```

- Per i dettagli sull'API, consulta la ImportCertificatesezione 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.

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListCerts {
    public static void main(String[] args) {
        listCertificates();
    }

    /**
     * Lists all the certificates managed by AWS Certificate Manager (ACM) that have
 a status of "ISSUED".
     */
    public static void listCertificates() {
        AcmClient acmClient = AcmClient.create();
        try {
            ListCertificatesRequest listRequest = ListCertificatesRequest.builder()
                .certificateStatuses(CertificateStatus.ISSUED)
                .maxItems(100)
                .build();
            ListCertificatesIterable listResponse =
 acmClient.listCertificatesPaginator(listRequest);
```

```
            // Print the certificate details using streams
            listResponse.certificateSummaryList().stream()
                .forEach(certificate -> {
                    System.out.println("Certificate ARN: " +
 certificate.certificateArn());
                    System.out.println("Certificate Domain Name: " +
 certificate.domainName());
                    System.out.println("Certificate Status: " +
 certificate.statusAsString());
                    System.out.println("---");
                });

        } catch (AcmException e) {
            System.err.println(e.getMessage());
        }
    }
}
```

- Per i dettagli sull'API, consulta la ListCertificatessezione AWS SDK for Java 2.x API Reference.

`ListTagsForCertificate`

Il seguente esempio di codice mostra come utilizzare`ListTagsForCertificate`.

SDK per Java 2.x

> **Note**
>
> C'è 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.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
```

```
public class ListCertTags {

    public static void main(String[] args) {

        final String usage = """

            Usage:    <certArn>

            Where:
                certArn - the ARN of the certificate.
            """;
        if (args.length != 1) {
            System.out.println(usage);
            return;
        }

        String certArn = args[0];
        listCertTags(certArn);
    }

    /**
     * Lists the tags associated with an AWS Certificate Manager (ACM) certificate.
     *
     * @param certArn the Amazon Resource Name (ARN) of the ACM certificate
     */
    public static void listCertTags(String certArn) {
        AcmClient acmClient = AcmClient.create();

        ListTagsForCertificateRequest request =
 ListTagsForCertificateRequest.builder()
            .certificateArn(certArn)
            .build();

        ListTagsForCertificateResponse response =
 acmClient.listTagsForCertificate(request);
        List<Tag> tagList = response.tags();
        tagList.forEach(tag -> {
            System.out.println("Key: " + tag.key());
            System.out.println("Value: " + tag.value());
        });
    }
}
```

- Per i dettagli sull'API, consulta la ListTagsForCertificatesezione AWS SDK for Java 2.x API Reference.

`RemoveTagsFromCertificate`

Il seguente esempio di codice mostra come utilizzare`RemoveTagsFromCertificate`.

SDK per Java 2.x

> **Note**
>
> C'è 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.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class RemoveTagsFromCert {

    public static void main(String[] args) {
        final String usage = """

            Usage:    <certArn>

            Where:
                certArn - the ARN of the certificate.
            """;
        if (args.length != 1) {
            System.out.println(usage);
            return;
        }

        String certArn = args[0];
        removeTags(certArn);
```

```
    }

    /**
     * Removes tags from an AWS Certificate Manager (ACM) certificate.
     *
     * @param certArn the Amazon Resource Name (ARN) of the certificate from which
 to remove tags
     */
    public static void removeTags(String certArn) {
        AcmClient acmClient = AcmClient.create();
        List<Tag> expectedTags =
 List.of(Tag.builder().key("key").value("value").build());
        RemoveTagsFromCertificateRequest req =
 RemoveTagsFromCertificateRequest.builder()
            .certificateArn(certArn)
            .tags(expectedTags)
            .build();

        try {
            acmClient.removeTagsFromCertificate(req);
            System.out.println("Successfully removed tags from the certificate");
        } catch (AcmException e) {
            System.err.println(e.getMessage());
        }
    }
}
```

- Per i dettagli sull'API, consulta la RemoveTagsFromCertificatesezione AWS SDK for Java 2.x API Reference.

`RenewCertificate`

Il seguente esempio di codice mostra come utilizzare`RenewCertificate`.

SDK per Java 2.x

> **Note**
>
> C'è 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.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class RenewCert {
    public static void main(String[] args) {
        final String usage = """

            Usage:    <certArn>

            Where:
                certArn - the ARN of the certificate.
            """;
        if (args.length != 1) {
            System.out.println(usage);
            return;
        }

        String certArn = args[0];
        renewCertificate(certArn);
    }

    /**
     * Renews an existing SSL/TLS certificate in AWS Certificate Manager (ACM).
     *
     * @param certArn The Amazon Resource Name (ARN) of the certificate to be
 renewed.
     * @throws AcmException If there is an error renewing the certificate.
     */
    public static void renewCertificate(String certArn) {
        AcmClient acmClient = AcmClient.create();

        RenewCertificateRequest certificateRequest =
 RenewCertificateRequest.builder()
            .certificateArn(certArn)
            .build();

        try {
```

```
            acmClient.renewCertificate(certificateRequest);
            System.out.println("The certificate was renewed");
        } catch(AcmException e){
            System.out.println(e.getMessage());
        }
    }
}
```

- Per i dettagli sull'API, consulta la RenewCertificatesezione AWS SDK for Java 2.x API Reference.

`RequestCertificate`

Il seguente esempio di codice mostra come utilizzare`RequestCertificate`.

SDK per Java 2.x

> **Note**
>
> C'è 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.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class RequestCert {

    public static void main(String[] args) {
        requestCertificate();
    }

    /**
     * Requests a certificate from the AWS Certificate Manager (ACM) service.
```

```
     */
    public static void requestCertificate() {
        AcmClient acmClient = AcmClient.create();
        ArrayList<String> san = new ArrayList<>();
        san.add("www.example.com");

        RequestCertificateRequest req = RequestCertificateRequest.builder()
            .domainName("example.com")
            .idempotencyToken("1Aq25pTy")
            .subjectAlternativeNames(san)
            .build();

        try {
            RequestCertificateResponse response = acmClient.requestCertificate(req);
            System.out.println("Cert ARN IS " + response.certificateArn());
        } catch (AcmException e) {
            System.err.println(e.getMessage());
        }
    }
}
```

- Per i dettagli sull'API, consulta la RequestCertificatesezione AWS SDK for Java 2.x API Reference.

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.

Gli scenari sono esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio o combinate con altri Servizi AWS.

AWS i contributi della community sono esempi che sono stati creati e gestiti da più team AWS. Per fornire feedback, utilizza il meccanismo fornito negli archivi collegati.

Ogni esempio include un collegamento al codice sorgente completo, dove puoi trovare istruzioni su come configurare ed eseguire il codice nel contesto.

Argomenti

- Azioni
- Scenari
- AWS contributi della comunità

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 CreateDeploymentsezione 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 CreateRestApisezione 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 DeleteDeploymentsezione 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 DeleteRestApisezione AWS SDK for Java 2.x API Reference.

Scenari

Creazione di un'applicazione serverless per gestire foto

Nell'esempio di codice seguente viene illustrato come creare un'applicazione serverless che consente agli utenti di gestire le foto mediante 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

Utilizzo di un'API Gateway per richiamare una funzione Lambda

Il seguente esempio di codice mostra come creare una AWS Lambda funzione richiamata da Amazon API Gateway.

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

AWS contributi della comunità

Crea e testa un'applicazione serverless

Il seguente esempio di codice mostra come creare e testare un'applicazione serverless utilizzando API Gateway con Lambda e DynamoDB

SDK per Java 2.x

> Mostra come creare e testare un'applicazione serverless composta da un API Gateway con Lambda e DynamoDB utilizzando Java SDK.
>
> Per il codice sorgente completo e le istruzioni su come configurarlo ed eseguirlo, consulta l'esempio completo su. GitHub
>
> Servizi utilizzati in questo esempio
>
> - API Gateway
> - DynamoDB
> - Lambda

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.

Ogni esempio include un collegamento al codice sorgente completo, in cui 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.ApplicationAutoScalingExceptio
import
 software.amazon.awssdk.services.applicationautoscaling.model.DeleteScalingPolicyRequest;
import
 software.amazon.awssdk.services.applicationautoscaling.model.DeregisterScalableTargetReques
import
 software.amazon.awssdk.services.applicationautoscaling.model.DescribeScalableTargetsRequest
import
 software.amazon.awssdk.services.applicationautoscaling.model.DescribeScalableTargetsRespons
import
 software.amazon.awssdk.services.applicationautoscaling.model.DescribeScalingPoliciesRequest
import
 software.amazon.awssdk.services.applicationautoscaling.model.DescribeScalingPoliciesRespons
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 DeleteScalingPolicysezione 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.ApplicationAutoScalingExceptio
import
 software.amazon.awssdk.services.applicationautoscaling.model.DescribeScalableTargetsRequest
import
 software.amazon.awssdk.services.applicationautoscaling.model.DescribeScalableTargetsRespons
import
 software.amazon.awssdk.services.applicationautoscaling.model.DescribeScalingPoliciesRequest
import
 software.amazon.awssdk.services.applicationautoscaling.model.DescribeScalingPoliciesRespons
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.TargetTrackingScalingPolicyCon
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 RegisterScalableTargetsezione 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.

Ogni esempio include un collegamento al codice sorgente completo, in cui 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 GetRoutingControlStatesezione 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 UpdateRoutingControlStatesezione 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 nozioni di base sono esempi di codice che mostrano come eseguire le operazioni essenziali all'interno di un servizio.

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.

Gli scenari sono esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio o combinate con altri Servizi AWS.

Ogni esempio include un collegamento al codice sorgente completo, in cui 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 i dettagli sull'API, consulta Descrivi DBClusters in AWS SDK for Java 2.x API Reference.

Argomenti

- Nozioni di base
- Azioni
- Scenari

Nozioni di base

Informazioni di base

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 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 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 .
  - CreaDBCluster
  - CreaDBClusterParameterGroup
  - Crea DBCluster istantanea
  - CreaDBInstance
  - EliminaDBCluster
  - EliminaDBClusterParameterGroup

- EliminaDBInstance
- Descriva DBCluster ParameterGroups
- DBClusterDescrivi parametri
- Descrivi le DBCluster istantanee
- Descriva DBClusters
- Descrivi DBEngine versioni
- Descriva DBInstances
- DescribeOrderableDBInstanceOpzioni
- ModificaDBClusterParameterGroup

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 i dettagli sull'API, consulta Create DBCluster in AWS SDK for Java 2.x API Reference.

`CreateDBClusterParameterGroup`

Il seguente esempio di codice mostra come utilizzare`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 Create DBCluster ParameterGroup 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 Create DBCluster Snapshot 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 i dettagli sull'API, consulta Create DBInstance in AWS SDK for Java 2.x API Reference.

`DeleteDBCluster`

Il seguente esempio di codice mostra come utilizzare`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 i dettagli sull'API, consulta Delete DBCluster in AWS SDK for Java 2.x API Reference.

`DeleteDBClusterParameterGroup`

Il seguente esempio di codice mostra come utilizzare`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 Delete DBCluster ParameterGroup 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 i dettagli sull'API, consulta Delete DBInstance in AWS SDK for Java 2.x API Reference.

`DescribeDBClusterParameterGroups`

Il seguente esempio di codice mostra come utilizzare`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 Descrivi DBCluster ParameterGroups 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 DBClusterDescrivi i parametri 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(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);
        }
    }
```

- Per i dettagli sull'API, consulta Descrivi le DBCluster istantanee 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 i dettagli sull'API, consulta Descrivi DBClusters in 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()
                    .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 Descrivi DBEngine le versioni 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 i dettagli sull'API, consulta Descrivi DBInstances in AWS SDK for Java 2.x API Reference.

`DescribeOrderableDBInstanceOptions`

Il seguente esempio di codice mostra come utilizzare`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 DescribeOrderableDBInstanceOpzioni 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 Modify DBCluster ParameterGroup in AWS SDK for Java 2.x API Reference.

Scenari

Creazione di un tracciatore di elementi di lavoro di Aurora Serverless

Il seguente esempio di codice mostra come creare un'applicazione Web che tiene traccia degli elementi di lavoro in un database Amazon Aurora Serverless e utilizza Amazon Simple Email Service (Amazon SES) per inviare report.

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

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 nozioni di base sono esempi di codice che mostrano come eseguire le operazioni essenziali all'interno di un servizio.

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.

Gli scenari sono esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio o combinate con altri Servizi AWS.

Ogni esempio include un collegamento al codice sorgente completo, in cui 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 DescribeAutoScalingGroupssezione AWS SDK for Java 2.x API Reference.

Argomenti

- Nozioni di base
- Azioni
- Scenari

Nozioni di base

Informazioni di base

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 = "Scott250" ; //rgs[0];
        String launchTemplateName = "MyTemplate5" ;//args[1];
        String vpcZoneId = "subnet-0ddc451b8a8a1aa44" ; //args[2];

        AutoScalingClient autoScalingClient = AutoScalingClient.builder()
                .region(Region.US_EAST_1)
                .build();

        Ec2Client ec2 = Ec2Client.create();

        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.numberOfAutoScalingGroups());

        } 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

Azioni

`CreateAutoScalingGroup`

Il seguente esempio di codice mostra come usare`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 CreateAutoScalingGroupsezione 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 DeleteAutoScalingGroupsezione 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 DescribeAutoScalingGroupssezione 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 DescribeAutoScalingInstancessezione 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 DescribeScalingActivitiessezione 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 DisableMetricsCollectionsezione 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 EnableMetricsCollectionsezione 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 SetDesiredCapacitysezione 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 TerminateInstanceInAutoScalingGroupsezione 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 UpdateAutoScalingGroupsezione 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 Amazon EC2 Auto Scaling per creare istanze Amazon Elastic Compute Cloud (Amazon EC2) basate su un modello di avvio e per mantenere il numero di istanze in un intervallo specificato.
- Gestisci e distribuisci 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 EC2 istanza 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 di integrità aggiornando AWS Systems Manager i parametri.

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.

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 wasSuccessul = loadBalancer.verifyLoadBalancerEndpoint(elbDnsName);
        if (!wasSuccessul) {
            System.out.println("Couldn't connect to the load balancer, verifying
that the port is open...");
            CloseableHttpClient httpClient = HttpClients.createDefault();

            // Create an HTTP GET request to "http://checkip.amazonaws.com"
            HttpGet httpGet = new HttpGet("http://checkip.amazonaws.com");
            try {
                // Execute the request and get the response
                HttpResponse response = httpClient.execute(httpGet);

                // Read the response content.
                String ipAddress =
IOUtils.toString(response.getEntity().getContent(), StandardCharsets.UTF_8).trim();

                // Print the public IP address.
                System.out.println("Public IP Address: " + ipAddress);
                GroupInfo groupInfo = autoScaler.verifyInboundPort(vpcId, port,
ipAddress);
                if (!groupInfo.isPortOpen()) {
                    System.out.println("""
                            For this example to work, the default security group for
your default VPC must
                            allow access from this computer. You can either add it
automatically from this
                            example or add it yourself using the AWS Management
Console.
                            """);

                    System.out.println(
                            "Do you want to add a rule to security group " +
groupInfo.getGroupName() + " to allow");
                    System.out.println("inbound traffic on port " + port + " from
your computer's IP address (y/n) ");
                    String ans = in.nextLine();
                    if ("y".equalsIgnoreCase(ans)) {
                        autoScaler.openInboundPort(groupInfo.getGroupName(),
String.valueOf(port), ipAddress);
                        System.out.println("Security group rule added.");
                    } else {
```

```
                        System.out.println("No security group rule added.");
                    }
                }

            } catch (AutoScalingException e) {
                e.printStackTrace();
            }
        } else if (wasSuccessul) {
            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 azioni di Auto Scaling e EC2 Amazon.

```
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.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)
                    .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

AWS Batch 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 Batch.

Le nozioni di base sono esempi di codice che mostrano come eseguire le operazioni essenziali all'interno di un servizio.

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.

Ogni esempio include un collegamento al codice sorgente completo, in cui è possibile trovare istruzioni su come configurare ed eseguire il codice nel contesto.

Nozioni di base

Salve AWS Batch

Il seguente esempio di codice mostra come iniziare a utilizzare AWS Batch.

SDK per Java 2.x

> **Note**
>
> C'è 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.core.retry.RetryPolicy;
import software.amazon.awssdk.http.async.SdkAsyncHttpClient;
import software.amazon.awssdk.http.nio.netty.NettyNioAsyncHttpClient;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.batch.BatchAsyncClient;
import software.amazon.awssdk.services.batch.model.JobStatus;
import software.amazon.awssdk.services.batch.model.JobSummary;
import software.amazon.awssdk.services.batch.model.ListJobsRequest;
import software.amazon.awssdk.services.batch.paginators.ListJobsPublisher;
import java.time.Duration;
import java.util.ArrayList;
import java.util.List;
```

```
import java.util.concurrent.CompletableFuture;

public class HelloBatch {
    private static BatchAsyncClient batchClient;

    public static void main(String[] args) {
        List<JobSummary> jobs = listJobs("my-job-queue");
        jobs.forEach(job ->
            System.out.printf("Job ID: %s, Job Name: %s, Job Status: %s%n",
                job.jobId(), job.jobName(), job.status())
        );
    }

    public static List<JobSummary> listJobs(String jobQueue) {
        if (jobQueue == null || jobQueue.isEmpty()) {
            throw new IllegalArgumentException("Job queue cannot be null or empty");
        }

        ListJobsRequest listJobsRequest = ListJobsRequest.builder()
            .jobQueue(jobQueue)
            .jobStatus(JobStatus.SUCCEEDED)
            .build();

        List<JobSummary> jobSummaries = new ArrayList<>();
        ListJobsPublisher listJobsPaginator =
 getAsyncClient().listJobsPaginator(listJobsRequest);
        CompletableFuture<Void> future = listJobsPaginator.subscribe(response -> {
            jobSummaries.addAll(response.jobSummaryList());
        });

        future.join();
        return jobSummaries;
    }

    private static BatchAsyncClient getAsyncClient() {
        SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
            .maxConcurrency(100)  // Increase max concurrency to handle more
 simultaneous connections.
            .connectionTimeout(Duration.ofSeconds(60))  // Set the connection
 timeout.
            .readTimeout(Duration.ofSeconds(60))  // Set the read timeout.
            .writeTimeout(Duration.ofSeconds(60))  // Set the write timeout.
            .build();
```

```
        ClientOverrideConfiguration overrideConfig =
 ClientOverrideConfiguration.builder()
            .apiCallTimeout(Duration.ofMinutes(2))  // Set the overall API call
 timeout.
            .apiCallAttemptTimeout(Duration.ofSeconds(90))  // Set the individual
 call attempt timeout.
            .retryPolicy(RetryPolicy.builder()  // Add a retry policy to handle
 transient errors.
                .numRetries(3)  // Number of retry attempts.
                .build())
            .build();

        if (batchClient == null) {
            batchClient = BatchAsyncClient.builder()
                .region(Region.US_EAST_1)
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return batchClient;
    }
}
```

- Per i dettagli sull'API, consulta la listJobsPaginatorsezione AWS SDK for Java 2.x API Reference.

Argomenti

- Nozioni di base
- Azioni

Nozioni di base

Informazioni di base

L'esempio di codice seguente mostra come:

- Crea un ambiente di AWS Batch calcolo.
- Controlla lo stato dell'ambiente di calcolo.
- Imposta una coda di AWS Batch lavoro e una definizione del lavoro.

- Registra una definizione di lavoro.
- Invia un AWS Batch Job.
- Ottieni un elenco di offerte di lavoro applicabili alla coda delle offerte di lavoro.
- Controlla lo stato del lavoro.
- Eliminare AWS Batch 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.

Esegui uno scenario interattivo che dimostri le AWS Batch funzionalità.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.batch.model.BatchException;
import software.amazon.awssdk.services.batch.model.ClientException;
import software.amazon.awssdk.services.batch.model.CreateComputeEnvironmentResponse;
import software.amazon.awssdk.services.batch.model.JobSummary;
import software.amazon.awssdk.services.ec2.Ec2AsyncClient;
import software.amazon.awssdk.services.ec2.model.DescribeSecurityGroupsRequest;
import software.amazon.awssdk.services.ec2.model.DescribeSecurityGroupsResponse;
import software.amazon.awssdk.services.ec2.model.DescribeSubnetsRequest;
import software.amazon.awssdk.services.ec2.model.DescribeSubnetsResponse;
import software.amazon.awssdk.services.ec2.model.DescribeVpcsRequest;
import software.amazon.awssdk.services.ec2.model.Filter;
import software.amazon.awssdk.services.ec2.model.SecurityGroup;
import software.amazon.awssdk.services.ec2.model.Subnet;
import software.amazon.awssdk.services.ec2.model.Vpc;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import java.util.List;
import java.util.Map;
import java.util.Scanner;
import java.util.UUID;
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.CompletionException;
```

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * NOTE
 * This scenario submits a job that pulls a Docker image named echo-text from Amazon
 ECR to Amazon Fargate.
 *
 * To place this Docker image on Amazon ECR, run the following Basics scenario.
 *
 * https://github.com/awsdocs/aws-doc-sdk-examples/tree/main/javav2/example_code/ecr
 *
 */
public class BatchScenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    // Define two stacks used in this Basics Scenario.
    private static final String ROLES_STACK = "RolesStack";
    private static String defaultSubnet;
    private static String defaultSecurityGroup;

    private static final Logger logger =
 LoggerFactory.getLogger(BatchScenario.class);

    public static void main(String[] args) throws InterruptedException {

        BatchActions batchActions = new BatchActions();
        Scanner scanner = new Scanner(System.in);
        String computeEnvironmentName = "my-compute-environment";
        String jobQueueName = "my-job-queue";
        String jobDefinitionName = "my-job-definition";


        // See the NOTE in this Java code example (at start).
        String dockerImage = "dkr.ecr.us-east-1.amazonaws.com/echo-text:echo-text";

        logger.info("""
            AWS Batch is a fully managed batch processing service that dynamically
 provisions the required compute
```

```
            resources for batch computing workloads. The Java V2 `BatchAsyncClient`
 allows
            developers to automate the submission, monitoring, and management of
 batch jobs.

            This scenario provides an example of setting up a compute environment,
 job queue and job definition,
            and then submitting a job.

            This scenario submits a job that pulls a Docker image named echo-text
 from Amazon ECR to Amazon Fargate.

            To place this Docker image on Amazon ECR, run the following Basics
 scenario.

            https://github.com/awsdocs/aws-doc-sdk-examples/tree/main/javav2/
example_code/ecr

            Let's get started...

            You have two choices:

            1 - Run the entire program.
            2 - Delete an existing Compute Environment (created from a previous
 execution of
            this program that did not complete).
            """);

        while (true) {
            String input = scanner.nextLine();
            if (input.trim().equalsIgnoreCase("1")) {
                logger.info("Continuing with the program...");
               // logger.info("");
                break;
            } else if (input.trim().equalsIgnoreCase("2")) {
                String jobQueueARN = String.valueOf(batchActions.
 describeJobQueueAsync(computeEnvironmentName));
                if (!jobQueueARN.isEmpty()) {
                    batchActions.disableJobQueueAsync(jobQueueARN);
                    countdown(1);
                    batchActions.deleteJobQueueAsync(jobQueueARN);
                }

                try {
```

```
batchActions.disableComputeEnvironmentAsync(computeEnvironmentName)
                        .exceptionally(ex -> {
                            logger.info("Disable compute environment failed: " +
ex.getMessage());
                            return null;
                        })
                        .join();
                } catch (CompletionException ex) {
                    logger.info("Failed to disable compute environment: " +
ex.getMessage());
                }
                countdown(2);

batchActions.deleteComputeEnvironmentAsync(computeEnvironmentName).join();
                return;
            } else {
                // Handle invalid input.
                logger.info("Invalid input. Please try again.");
            }
        }
        System.out.println(DASHES);

        waitForInputToContinue(scanner);
        // Get an AWS Account id used to retrieve the docker image from Amazon ECR.
        // Create a single-element array to store the `accountId` value.
        String[] accId = new String[1];
        CompletableFuture<String> accountIdFuture = batchActions.getAccountId();
        accountIdFuture.thenAccept(accountId -> {
            logger.info("Account ID: " + accountId);
            accId[0] = accountId;
        }).join();

        dockerImage = accId[0]+"."+dockerImage;

        // Get a default subnet and default security associated with the default
VPC.
        getSubnetSecurityGroup();

        logger.info("Use AWS CloudFormation to create two IAM roles that are
required for this scenario.");
        CloudFormationHelper.deployCloudFormationStack(ROLES_STACK);
```

```
        Map<String, String> stackOutputs =
CloudFormationHelper.getStackOutputs(ROLES_STACK);
        String batchIAMRole = stackOutputs.get("BatchRoleArn");
        String executionRoleARN = stackOutputs.get("EcsRoleArn");

        logger.info("The IAM role needed to interact with AWS Batch is
"+batchIAMRole);
        waitForInputToContinue(scanner);

        logger.info(DASHES);
        logger.info("1. Create a Batch compute environment");
        logger.info("""
            A compute environment is a resource where you can run your batch jobs.
            After creating a compute environment, you can define job queues and job
definitions to submit jobs for
            execution.

            The benefit of creating a compute environment is it allows you to easily
configure and manage the compute
            resources that will be used to run your Batch jobs. By separating the
compute environment from the job definitions,
            you can easily scale your compute resources up or down as needed,
without having to modify your job definitions.
            This makes it easier to manage your Batch workloads and ensures that
your jobs have the necessary
            compute resources to run efficiently.
            """);

        waitForInputToContinue(scanner);
        try {
            CompletableFuture<CreateComputeEnvironmentResponse> future =
batchActions.createComputeEnvironmentAsync(computeEnvironmentName, batchIAMRole,
defaultSubnet, defaultSecurityGroup);
            CreateComputeEnvironmentResponse response = future.join();
            logger.info("Compute Environment ARN: " +
response.computeEnvironmentArn());
        } catch (RuntimeException rte) {
            Throwable cause = rte.getCause();
            if (cause instanceof ClientException batchExceptionEx) {
                String myErrorCode =
batchExceptionEx.awsErrorDetails().errorMessage();
                if ("Object already exists".contains(myErrorCode)) {
                    logger.info("The compute environment '" + computeEnvironmentName
+ "' already exists. Moving on...");
```

```
                } else {
                    logger.info("Batch error occurred: {} (Code: {})",
batchExceptionEx.getMessage(), batchExceptionEx.awsErrorDetails().errorCode());
                    return;
                }
            } else {
                    logger.info("An unexpected error occurred: {}", (cause != null ?
cause.getMessage() : rte.getMessage()));
            }
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("2. Check the status of the "+computeEnvironmentName +" Compute
Environment.");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<String> future =
batchActions.checkComputeEnvironmentsStatus(computeEnvironmentName);
            String status = future.join();
            logger.info("Compute Environment Status: " + status);

        } catch (RuntimeException rte) {
            Throwable cause = rte.getCause();
            if (cause instanceof ClientException batchExceptionEx) {
                logger.info("Batch error occurred: {} (Code: {})",
batchExceptionEx.getMessage(), batchExceptionEx.awsErrorDetails().errorCode());
                return;
            } else {
                logger.info("An unexpected error occurred: " + (cause != null ?
cause.getMessage() : rte.getMessage()));
                return;
            }
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("3. Create a job queue");
        logger.info("""
             A job queue is an essential component that helps manage the execution
of your batch jobs.
```

```
            It acts as a buffer, where jobs are placed and then scheduled for
execution based on their
            priority and the available resources in the compute environment.
            """);
        waitForInputToContinue(scanner);

        String jobQueueArn = null;
        try {
            CompletableFuture<String> jobQueueFuture =
batchActions.createJobQueueAsync(jobQueueName, computeEnvironmentName);
            jobQueueArn = jobQueueFuture.join();
            logger.info("Job Queue ARN: " + jobQueueArn);

        } catch (RuntimeException rte) {
            Throwable cause = rte.getCause();
            if (cause instanceof BatchException batchExceptionEx) {
                String myErrorCode =
batchExceptionEx.awsErrorDetails().errorMessage();
                if ("Object already exists".contains(myErrorCode)) {
                    logger.info("The job queue '" + jobQueueName + "' already
exists. Moving on...");
                    // Retrieve the ARN of the job queue.
                    CompletableFuture<String> jobQueueArnFuture =
batchActions.getJobQueueARN(jobQueueName);
                    jobQueueArn = jobQueueArnFuture.join();
                    logger.info("Job Queue ARN: " + jobQueueArn);
                } else {
                    logger.info("Batch error occurred: {} (Code: {})",
batchExceptionEx.getMessage(), batchExceptionEx.awsErrorDetails().errorCode());
                    return;
                }
            } else {
                logger.info("An unexpected error occurred: " + (cause != null ?
cause.getMessage() : rte.getMessage()));
                return; // End the execution
            }
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info("4. Register a Job Definition.");
        logger.info("""
            Registering a job in AWS Batch using the Fargate launch type ensures
that all
```

```
            necessary parameters, such as the execution role, command to run, and so
on
            are specified and reused across multiple job submissions.

             The job definition pulls a Docker image from Amazon ECR and executes
the Docker image.
            """);

        waitForInputToContinue(scanner);
        String jobARN;
        try {
            String platform = "";
            while (true) {
                logger.info("""
                    On which platform/CPU architecture combination did you build the
Docker image?:
                    1. Windows       X86_64
                    2. Mac or Linux  ARM64
                    3. Mac or Linux  X86_64

                    Please select 1, 2, or 3.
                    """);
                String platAns = scanner.nextLine().trim();
                if (platAns.equals("1")) {
                    platform = "X86_64";
                    break; // Exit loop since a valid option is selected
                } else if (platAns.equals("2")) {
                    platform = "ARM64";
                    break; // Exit loop since a valid option is selected
                } else if (platAns.equals("3")) {
                    platform = "X86_64";
                    break; // Exit loop since a valid option is selected
                } else {
                    System.out.println("Invalid input. Please select either 1 or
2.");
                }
            }

            jobARN = batchActions.registerJobDefinitionAsync(jobDefinitionName,
executionRoleARN, dockerImage, platform)
                .exceptionally(ex -> {
                    System.err.println("Register job definition failed: " +
ex.getMessage());
                    return null;
```

```
                })
                .join();
            if (jobARN != null) {
                logger.info("Job ARN: " + jobARN);
            }
        } catch (RuntimeException rte) {
            logger.error("A Batch exception occurred while registering the job: {}",
rte.getCause() != null ? rte.getCause().getMessage() : rte.getMessage());
            return;
        }
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("5. Submit an AWS Batch job from a job definition.");
        waitForInputToContinue(scanner);
        String jobId;
        try {
            jobId = batchActions.submitJobAsync(jobDefinitionName, jobQueueName,
jobARN)
                .exceptionally(ex -> {
                    System.err.println("Submit job failed: " + ex.getMessage());
                    return null;
                })
                .join();

            logger.info("The job id is "+jobId);
            logger.info("Let's wait 2 minutes for the job to complete");
            countdown(2);

        } catch (RuntimeException rte) {
            logger.error("A Batch exception occurred while submitting the job: {}",
rte.getCause() != null ? rte.getCause().getMessage() : rte.getMessage());
            return;
        }
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        logger.info(DASHES);
        logger.info("6. Get a list of jobs applicable to the job queue.");

        waitForInputToContinue(scanner);
        try {
            List<JobSummary> jobs = batchActions.listJobsAsync(jobQueueName);
            jobs.forEach(job ->
```

```
                logger.info("Job ID: {}, Job Name: {}, Job Status: {}", job.jobId(),
job.jobName(), job.status()));

        } catch (RuntimeException rte) {
            logger.info("A Batch exception occurred while submitting the job: {}",
rte.getCause() != null ? rte.getCause().getMessage() : rte.getMessage());
            return;
        }

        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("7. Check the status of job "+jobId);
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<String> future = batchActions.describeJobAsync(jobId);
            String jobStatus = future.join();
            logger.info("Job Status: " + jobStatus);

        } catch (RuntimeException rte) {
            logger.info("A Batch exception occurred while submitting the job: {}",
rte.getCause() != null ? rte.getCause().getMessage() : rte.getMessage());
            return;
        }

        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        logger.info("8. Delete Batch resources");
        logger.info(
            """
            When deleting an AWS Batch compute environment, it does not happen
instantaneously.
            There is typically a delay, similar to some other AWS resources.
            AWS Batch starts the deletion process.
            """);
        logger.info("Would you like to delete the AWS Batch resources such as the
compute environment? (y/n)");
        String delAns = scanner.nextLine().trim();
        if (delAns.equalsIgnoreCase("y")) {
            logger.info("You selected to delete the AWS ECR resources.");
            logger.info("First, we will deregister the Job Definition.");
            waitForInputToContinue(scanner);
```

```
            try {
                batchActions.deregisterJobDefinitionAsync(jobARN)
                    .exceptionally(ex -> {
                        logger.info("Deregister job definition failed: " +
ex.getMessage());
                        return null;
                    })
                    .join();
                logger.info(jobARN + " was deregistered");
            } catch (RuntimeException rte) {
                logger.error("A Batch exception occurred: {}", rte.getCause() !=
null ? rte.getCause().getMessage() : rte.getMessage());
                return;
            }

            logger.info("Second, we will disable and then delete the Job Queue.");
            waitForInputToContinue(scanner);
            try {
                batchActions.disableJobQueueAsync(jobQueueArn)
                    .exceptionally(ex -> {
                        logger.info("Disable job queue failed: " + ex.getMessage());
                        return null;
                    })
                    .join();
                logger.info(jobQueueArn + " was disabled");
            } catch (RuntimeException rte) {
                logger.info("A Batch exception occurred: {}", rte.getCause() !=
null ? rte.getCause().getMessage() : rte.getMessage());
                return;
            }

            batchActions.waitForJobQueueToBeDisabledAsync(jobQueueArn);
            try {
                CompletableFuture<Void> future =
batchActions.waitForJobQueueToBeDisabledAsync(jobQueueArn);
                future.join();
                logger.info("Job queue is now disabled.");
            } catch (RuntimeException rte) {
                logger.info("A Batch exception occurred: {}", rte.getCause() !=
null ? rte.getCause().getMessage() : rte.getMessage());
                return;
            }

            waitForInputToContinue(scanner);
```

```
            try {
                batchActions.deleteJobQueueAsync(jobQueueArn);
                logger.info(jobQueueArn +" was deleted");
            } catch (RuntimeException rte) {
                logger.info("A Batch exception occurred: {}", rte.getCause() !=
null ? rte.getCause().getMessage() : rte.getMessage());
                return;
            }
            logger.info("Let's wait 2 minutes for the job queue to be deleted");
            countdown(2);
            waitForInputToContinue(scanner);

            logger.info("Third, we will delete the Compute Environment.");
            waitForInputToContinue(scanner);
            try {
                batchActions.disableComputeEnvironmentAsync(computeEnvironmentName)
                    .exceptionally(ex -> {
                        System.err.println("Disable compute environment failed: " +
ex.getMessage());
                        return null;
                    })
                    .join();
                logger.info("Compute environment disabled") ;
            } catch (RuntimeException rte) {
                logger.info("A Batch exception occurred: {}", rte.getCause() !=
null ? rte.getCause().getMessage() : rte.getMessage());
                return;
            }


batchActions.checkComputeEnvironmentsStatus(computeEnvironmentName).thenAccept(state
-> {
                logger.info("Current State: " + state);
            }).join();

            logger.info("Lets wait 1 min for the compute environment to be
deleted");
            countdown(1);

            try {

batchActions.deleteComputeEnvironmentAsync(computeEnvironmentName).join();
                logger.info(computeEnvironmentName +" was deleted.");
```

```
            } catch (RuntimeException rte) {
                logger.info("A Batch exception occurred: {}", rte.getCause() !=
null ? rte.getCause().getMessage() : rte.getMessage());
                return;
            }
            waitForInputToContinue(scanner);
            CloudFormationHelper.destroyCloudFormationStack(ROLES_STACK);
        }

        logger.info(DASHES);
        logger.info("This concludes the AWS Batch SDK scenario");
        logger.info(DASHES);
    }

    private static void waitForInputToContinue(Scanner scanner) {
        while (true) {
            logger.info("");
            logger.info("Enter 'c' followed by <ENTER> to continue:");
            String input = scanner.nextLine();

            if (input.trim().equalsIgnoreCase("c")) {
                logger.info("Continuing with the program...");
                logger.info("");
                break;
            } else {
                // Handle invalid input.
                logger.info("Invalid input. Please try again.");
            }
        }
    }

    public static void countdown(int minutes) throws InterruptedException {
        int seconds = 0;
        for (int i = minutes * 60 + seconds; i >= 0; i--) {
            int displayMinutes = i / 60;
            int displaySeconds = i % 60;
            System.out.print(String.format("\r%02d:%02d", displayMinutes,
displaySeconds));
            Thread.sleep(1000); // Wait for 1 second
        }
        logger.info("Countdown complete!");
    }

    private static void getSubnetSecurityGroup() {
```

```
        try (Ec2AsyncClient ec2Client = Ec2AsyncClient.create()) {
            CompletableFuture<Vpc> defaultVpcFuture =
ec2Client.describeVpcs(DescribeVpcsRequest.builder()
                            .filters(Filter.builder()
                                    .name("is-default")
                                    .values("true")
                                    .build())
                            .build())
                    .thenApply(response -> response.vpcs().stream()
                            .findFirst()
                            .orElseThrow(() -> new RuntimeException("Default VPC not
found")));

            CompletableFuture<String> defaultSubnetFuture = defaultVpcFuture
                    .thenCompose(vpc ->
ec2Client.describeSubnets(DescribeSubnetsRequest.builder()
                                    .filters(Filter.builder()
                                                    .name("vpc-id")
                                                    .values(vpc.vpcId())
                                                    .build(),
                                            Filter.builder()
                                                    .name("default-for-az")
                                                    .values("true")
                                                    .build())
                                    .build())
                            .thenApply(DescribeSubnetsResponse::subnets)
                            .thenApply(subnets -> subnets.stream()
                                    .findFirst()
                                    .map(Subnet::subnetId)
                                    .orElseThrow(() -> new RuntimeException("No
default subnet found"))));

            CompletableFuture<String> defaultSecurityGroupFuture = defaultVpcFuture
                    .thenCompose(vpc ->
ec2Client.describeSecurityGroups(DescribeSecurityGroupsRequest.builder()
                                    .filters(Filter.builder()
                                                    .name("group-name")
                                                    .values("default")
                                                    .build(),
                                            Filter.builder()
                                                    .name("vpc-id")
                                                    .values(vpc.vpcId())
                                                    .build())
                                    .build())
```

```
 .thenApply(DescribeSecurityGroupsResponse::securityGroups)
                            .thenApply(securityGroups -> securityGroups.stream()
                                    .findFirst()
                                    .map(SecurityGroup::groupId)
                                    .orElseThrow(() -> new RuntimeException("No
 default security group found"))));

            defaultSubnet = defaultSubnetFuture.join();
            defaultSecurityGroup = defaultSecurityGroupFuture.join();
        }
    }
}
```

Una classe wrapper per i metodi SDK. AWS Batch

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.client.config.ClientOverrideConfiguration;
import software.amazon.awssdk.core.retry.RetryPolicy;
import software.amazon.awssdk.http.async.SdkAsyncHttpClient;
import software.amazon.awssdk.http.nio.netty.NettyNioAsyncHttpClient;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.batch.BatchAsyncClient;
import software.amazon.awssdk.services.batch.BatchClient;
import software.amazon.awssdk.services.batch.model.AssignPublicIp;
import software.amazon.awssdk.services.batch.model.BatchException;
import software.amazon.awssdk.services.batch.model.CEState;
import software.amazon.awssdk.services.batch.model.CEType;
import software.amazon.awssdk.services.batch.model.CRType;
import software.amazon.awssdk.services.batch.model.ComputeEnvironmentOrder;
import software.amazon.awssdk.services.batch.model.ComputeResource;
import software.amazon.awssdk.services.batch.model.ContainerProperties;
import software.amazon.awssdk.services.batch.model.CreateComputeEnvironmentRequest;
import software.amazon.awssdk.services.batch.model.CreateComputeEnvironmentResponse;
import software.amazon.awssdk.services.batch.model.CreateJobQueueRequest;
import software.amazon.awssdk.services.batch.model.DeleteComputeEnvironmentRequest;
import software.amazon.awssdk.services.batch.model.DeleteComputeEnvironmentResponse;
import software.amazon.awssdk.services.batch.model.DeleteJobQueueRequest;
import software.amazon.awssdk.services.batch.model.DeleteJobQueueResponse;
import software.amazon.awssdk.services.batch.model.DeregisterJobDefinitionRequest;
import software.amazon.awssdk.services.batch.model.DeregisterJobDefinitionResponse;
```

```
import
 software.amazon.awssdk.services.batch.model.DescribeComputeEnvironmentsRequest;
import
 software.amazon.awssdk.services.batch.model.DescribeComputeEnvironmentsResponse;
import software.amazon.awssdk.services.batch.model.DescribeJobQueuesRequest;
import software.amazon.awssdk.services.batch.model.DescribeJobQueuesResponse;
import software.amazon.awssdk.services.batch.model.DescribeJobsRequest;
import software.amazon.awssdk.services.batch.model.DescribeJobsResponse;
import software.amazon.awssdk.services.batch.model.JQState;
import software.amazon.awssdk.services.batch.model.JobDefinitionType;
import software.amazon.awssdk.services.batch.model.JobDetail;
import software.amazon.awssdk.services.batch.model.JobQueueDetail;
import software.amazon.awssdk.services.batch.model.JobStatus;
import software.amazon.awssdk.services.batch.model.JobSummary;
import software.amazon.awssdk.services.batch.model.ListJobsRequest;
import software.amazon.awssdk.services.batch.model.RegisterJobDefinitionResponse;
import software.amazon.awssdk.services.batch.model.NetworkConfiguration;
import software.amazon.awssdk.services.batch.model.PlatformCapability;
import software.amazon.awssdk.services.batch.model.RegisterJobDefinitionRequest;
import software.amazon.awssdk.services.batch.model.ResourceRequirement;
import software.amazon.awssdk.services.batch.model.ResourceType;
import software.amazon.awssdk.services.batch.model.RuntimePlatform;
import software.amazon.awssdk.services.batch.model.SubmitJobRequest;
import software.amazon.awssdk.services.batch.model.CreateJobQueueResponse;
import java.time.Duration;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Collections;
import java.util.List;
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.atomic.AtomicBoolean;
import
 software.amazon.awssdk.auth.credentials.EnvironmentVariableCredentialsProvider;
import software.amazon.awssdk.services.batch.model.SubmitJobResponse;
import software.amazon.awssdk.services.batch.model.UpdateComputeEnvironmentRequest;
import software.amazon.awssdk.services.batch.model.UpdateComputeEnvironmentResponse;
import software.amazon.awssdk.services.batch.model.UpdateJobQueueRequest;
import software.amazon.awssdk.services.batch.model.UpdateJobQueueResponse;
import software.amazon.awssdk.services.batch.paginators.ListJobsPublisher;
import software.amazon.awssdk.services.sts.StsAsyncClient;
import software.amazon.awssdk.services.sts.model.GetCallerIdentityResponse;

public class BatchActions {
    private static BatchAsyncClient batchClient;
```

```
   private static final Logger logger =
LoggerFactory.getLogger(BatchActions.class);

   private static BatchAsyncClient getAsyncClient() {
       if (batchClient == null) {
           SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
               .maxConcurrency(100)
               .connectionTimeout(Duration.ofSeconds(60))
               .readTimeout(Duration.ofSeconds(60))
               .writeTimeout(Duration.ofSeconds(60))
               .build();

           ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
               .apiCallTimeout(Duration.ofMinutes(2))
               .apiCallAttemptTimeout(Duration.ofSeconds(90))
               .retryPolicy(RetryPolicy.builder()
                   .numRetries(3)
                   .build())
               .build();

           batchClient = BatchAsyncClient.builder()
               .region(Region.US_EAST_1)
               .httpClient(httpClient)
               .overrideConfiguration(overrideConfig)
               .build();
       }
       return batchClient;
   }

   /**
    * Asynchronously creates a new compute environment in AWS Batch.
    *
    * @param computeEnvironmentName the name of the compute environment to create
    * @param batchIAMRole the IAM role to be used by the compute environment
    * @param subnet the subnet ID to be used for the compute environment
    * @param secGroup the security group ID to be used for the compute environment
    * @return a {@link CompletableFuture} representing the asynchronous operation,
which will complete with the
    *         {@link CreateComputeEnvironmentResponse} when the compute environment
has been created
    * @throws BatchException if there is an error creating the compute environment
    * @throws RuntimeException if there is an unexpected error during the operation
```

```
     */
    public CompletableFuture<CreateComputeEnvironmentResponse>
createComputeEnvironmentAsync(
        String computeEnvironmentName, String batchIAMRole, String subnet, String
secGroup) {
        CreateComputeEnvironmentRequest environmentRequest =
CreateComputeEnvironmentRequest.builder()
            .computeEnvironmentName(computeEnvironmentName)
            .type(CEType.MANAGED)
            .state(CEState.ENABLED)
            .computeResources(ComputeResource.builder()
                .type(CRType.FARGATE)
                .maxvCpus(256)
                .subnets(Collections.singletonList(subnet))
                .securityGroupIds(Collections.singletonList(secGroup))
                .build())
            .serviceRole(batchIAMRole)
            .build();

        CompletableFuture<CreateComputeEnvironmentResponse> response =
getAsyncClient().createComputeEnvironment(environmentRequest);
        response.whenComplete((resp, ex) -> {
            if (ex != null) {
               String errorMessage = "Unexpected error occurred: " +
ex.getMessage();
               throw new RuntimeException(errorMessage, ex);
            }
        });

        return response;
    }

    public CompletableFuture<DeleteComputeEnvironmentResponse>
deleteComputeEnvironmentAsync(String computeEnvironmentName) {
        DeleteComputeEnvironmentRequest deleteComputeEnvironment =
DeleteComputeEnvironmentRequest.builder()
            .computeEnvironment(computeEnvironmentName)
            .build();

        return getAsyncClient().deleteComputeEnvironment(deleteComputeEnvironment)
            .whenComplete((response, ex) -> {
                if (ex != null) {
                    Throwable cause = ex.getCause();
                    if (cause instanceof BatchException) {
```

```
                        throw new RuntimeException(cause);
                    } else {
                        throw new RuntimeException("Unexpected error: " +
cause.getMessage(), cause);
                    }
                }
            });
    }

    /**
     * Checks the status of the specified compute environment.
     *
     * @param computeEnvironmentName the name of the compute environment to check
     * @return a CompletableFuture containing the status of the compute environment,
or "ERROR" if an exception occurs
     */
    public CompletableFuture<String> checkComputeEnvironmentsStatus(String
computeEnvironmentName) {
        if (computeEnvironmentName == null || computeEnvironmentName.isEmpty()) {
            throw new IllegalArgumentException("Compute environment name cannot be
null or empty");
        }

        DescribeComputeEnvironmentsRequest environmentsRequest =
DescribeComputeEnvironmentsRequest.builder()
            .computeEnvironments(computeEnvironmentName)
            .build();

        CompletableFuture<DescribeComputeEnvironmentsResponse> response =
getAsyncClient().describeComputeEnvironments(environmentsRequest);
        response.whenComplete((resp, ex) -> {
            if (ex != null) {
                String errorMessage = "Unexpected error occurred: " +
ex.getMessage();
                throw new RuntimeException(errorMessage, ex);
            }
        });

        return response.thenApply(resp -> resp.computeEnvironments().stream()
            .map(env -> env.statusAsString())
            .findFirst()
            .orElse("UNKNOWN"));
    }
```

```
   /**
    * Creates a job queue asynchronously.
    *
    * @param jobQueueName the name of the job queue to create
    * @param computeEnvironmentName the name of the compute environment to
associate with the job queue
    * @return a CompletableFuture that completes with the Amazon Resource Name
(ARN) of the job queue
    */
   public CompletableFuture<String> createJobQueueAsync(String jobQueueName, String
computeEnvironmentName) {
       if (jobQueueName == null || jobQueueName.isEmpty()) {
           throw new IllegalArgumentException("Job queue name cannot be null or
empty");
       }
       if (computeEnvironmentName == null || computeEnvironmentName.isEmpty()) {
           throw new IllegalArgumentException("Compute environment name cannot be
null or empty");
       }

       CreateJobQueueRequest request = CreateJobQueueRequest.builder()
           .jobQueueName(jobQueueName)
           .priority(1)
           .computeEnvironmentOrder(ComputeEnvironmentOrder.builder()
               .computeEnvironment(computeEnvironmentName)
               .order(1)
               .build())
           .build();

       CompletableFuture<CreateJobQueueResponse> response =
getAsyncClient().createJobQueue(request);
       response.whenComplete((resp, ex) -> {
           if (ex != null) {
               String errorMessage = "Unexpected error occurred: " +
ex.getMessage();
               throw new RuntimeException(errorMessage, ex);
           }
       });

       return response.thenApply(CreateJobQueueResponse::jobQueueArn);
   }

   /**
```

```
    * Asynchronously lists the jobs in the specified job queue with the given job
status.
    *
    * @param jobQueue the name of the job queue to list jobs from
    * @return a List<JobSummary> that contains the jobs that succeeded
    */
   public List<JobSummary> listJobsAsync(String jobQueue) {
       if (jobQueue == null || jobQueue.isEmpty()) {
           throw new IllegalArgumentException("Job queue cannot be null or empty");
       }

       ListJobsRequest listJobsRequest = ListJobsRequest.builder()
           .jobQueue(jobQueue)
           .jobStatus(JobStatus.SUCCEEDED)  // Filter jobs by status.
           .build();

       List<JobSummary> jobSummaries = new ArrayList<>();
       ListJobsPublisher listJobsPaginator =
getAsyncClient().listJobsPaginator(listJobsRequest);
       CompletableFuture<Void> future = listJobsPaginator.subscribe(response -> {
           jobSummaries.addAll(response.jobSummaryList());
       });
       future.join();
       return jobSummaries;
   }

   /**
    * Registers a new job definition asynchronously in AWS Batch.
    * <p>
    * When using Fargate as the compute environment, it is crucial to set the
    * {@link NetworkConfiguration} with {@link AssignPublicIp#ENABLED} to
    * ensure proper networking configuration for the Fargate tasks. This
    * allows the tasks to communicate with external services, access the
    * internet, or communicate within a VPC.
    *
    * @param jobDefinitionName the name of the job definition to be registered
    * @param executionRoleARN the ARN (Amazon Resource Name) of the execution role
    *                         that provides permissions for the containers in the
job
    * @param cpuArch a value of either X86_64 or ARM64 required for the service
call
    * @return a CompletableFuture that completes with the ARN of the registered
    *         job definition upon successful execution, or completes exceptionally
with
```

```
     *         an error if the registration fails
     */
    public CompletableFuture<String> registerJobDefinitionAsync(String
jobDefinitionName, String executionRoleARN, String image, String cpuArch) {
        NetworkConfiguration networkConfiguration = NetworkConfiguration.builder()
                .assignPublicIp(AssignPublicIp.ENABLED)
                .build();

        ContainerProperties containerProperties = ContainerProperties.builder()
                .image(image)
                .executionRoleArn(executionRoleARN)
                .resourceRequirements(
                        Arrays.asList(
                                ResourceRequirement.builder()
                                        .type(ResourceType.VCPU)
                                        .value("1")
                                        .build(),
                                ResourceRequirement.builder()
                                        .type(ResourceType.MEMORY)
                                        .value("2048")
                                        .build()
                        )
                )
                .networkConfiguration(networkConfiguration)
                .runtimePlatform(b -> b
                        .cpuArchitecture(cpuArch)
                        .operatingSystemFamily("LINUX"))
                .build();

        RegisterJobDefinitionRequest request =
RegisterJobDefinitionRequest.builder()
                .jobDefinitionName(jobDefinitionName)
                .type(JobDefinitionType.CONTAINER)
                .containerProperties(containerProperties)
                .platformCapabilities(PlatformCapability.FARGATE)
                .build();

        CompletableFuture<String> future = new CompletableFuture<>();
        getAsyncClient().registerJobDefinition(request)
                .thenApply(RegisterJobDefinitionResponse::jobDefinitionArn)
                .whenComplete((result, ex) -> {
                    if (ex != null) {
                        future.completeExceptionally(ex);
                    } else {
```

```
                        future.complete(result);
                    }
                });

        return future;
    }

    /**
     * Deregisters a job definition asynchronously.
     *
     * @param jobDefinition the name of the job definition to be deregistered
     * @return a CompletableFuture that completes when the job definition has been
deregistered
     * or an exception has occurred
     */
    public CompletableFuture<DeregisterJobDefinitionResponse>
deregisterJobDefinitionAsync(String jobDefinition) {
        DeregisterJobDefinitionRequest jobDefinitionRequest =
DeregisterJobDefinitionRequest.builder()
            .jobDefinition(jobDefinition)
            .build();

        CompletableFuture<DeregisterJobDefinitionResponse> responseFuture =
getAsyncClient().deregisterJobDefinition(jobDefinitionRequest);
        responseFuture.whenComplete((response, ex) -> {
            if (ex != null) {
                throw new RuntimeException("Unexpected error occurred: " +
ex.getMessage(), ex);
            }
        });

        return responseFuture;
    }

    /**
     * Disables the specified job queue asynchronously.
     *
     * @param jobQueueArn the Amazon Resource Name (ARN) of the job queue to be
disabled
     * @return a {@link CompletableFuture} that completes when the job queue update
operation is complete,
     *         or completes exceptionally if an error occurs during the operation
     */
    public CompletableFuture<Void> disableJobQueueAsync(String jobQueueArn) {
```

```
        UpdateJobQueueRequest updateRequest = UpdateJobQueueRequest.builder()
            .jobQueue(jobQueueArn)
            .state(JQState.DISABLED)
            .build();

        CompletableFuture<UpdateJobQueueResponse> responseFuture =
getAsyncClient().updateJobQueue(updateRequest);
        return responseFuture.whenComplete((updateResponse, ex) -> {
            if (ex != null) {
                throw new RuntimeException("Failed to update job queue: " +
ex.getMessage(), ex);
            }
        }).thenApply(updateResponse -> null);
    }

    /**
     * Deletes a Batch job queue asynchronously.
     *
     * @param jobQueueArn The Amazon Resource Name (ARN) of the job queue to delete.
     * @return A CompletableFuture that represents the asynchronous deletion of the
job queue.
     *         The future completes when the job queue has been successfully deleted
or if an error occurs.
     *         If successful, the future will be completed with a {@code Void}
value.
     *         If an error occurs, the future will be completed exceptionally with
the thrown exception.
     */
    public CompletableFuture<Void> deleteJobQueueAsync(String jobQueueArn) {
        DeleteJobQueueRequest deleteRequest = DeleteJobQueueRequest.builder()
            .jobQueue(jobQueueArn)
            .build();

        CompletableFuture<DeleteJobQueueResponse> responseFuture =
getAsyncClient().deleteJobQueue(deleteRequest);
        return responseFuture.whenComplete((deleteResponse, ex) -> {
            if (ex != null) {
                throw new RuntimeException("Failed to delete job queue: " +
ex.getMessage(), ex);
            }
        }).thenApply(deleteResponse -> null);
    }

    /**
```

```
    * Asynchronously describes the job queue associated with the specified compute
environment.
    *
    * @param computeEnvironmentName the name of the compute environment to find the
associated job queue for
    * @return a {@link CompletableFuture} that, when completed, contains the job
queue ARN associated with the specified compute environment
    * @throws RuntimeException if the job queue description fails
    */
   public CompletableFuture<String> describeJobQueueAsync(String
computeEnvironmentName) {
       DescribeJobQueuesRequest describeJobQueuesRequest =
DescribeJobQueuesRequest.builder()
           .build();

       CompletableFuture<DescribeJobQueuesResponse> responseFuture =
getAsyncClient().describeJobQueues(describeJobQueuesRequest);
       return responseFuture.whenComplete((describeJobQueuesResponse, ex) -> {
           if (describeJobQueuesResponse != null) {
               String jobQueueARN;
               for (JobQueueDetail jobQueueDetail :
describeJobQueuesResponse.jobQueues()) {
                   for (ComputeEnvironmentOrder computeEnvironmentOrder :
jobQueueDetail.computeEnvironmentOrder()) {
                       String computeEnvironment =
computeEnvironmentOrder.computeEnvironment();
                       String name = getComputeEnvironmentName(computeEnvironment);
                       if (name.equals(computeEnvironmentName)) {
                           jobQueueARN = jobQueueDetail.jobQueueArn();
                           logger.info("Job queue ARN associated with the compute
environment: " + jobQueueARN);
                       }
                   }
               }
           } else {
               throw new RuntimeException("Failed to describe job queue: " +
ex.getMessage(), ex);
           }
       }).thenApply(describeJobQueuesResponse -> {
           String jobQueueARN = "";
           for (JobQueueDetail jobQueueDetail :
describeJobQueuesResponse.jobQueues()) {
               for (ComputeEnvironmentOrder computeEnvironmentOrder :
jobQueueDetail.computeEnvironmentOrder()) {
```

```
                    String computeEnvironment =
computeEnvironmentOrder.computeEnvironment();
                    String name = getComputeEnvironmentName(computeEnvironment);
                    if (name.equals(computeEnvironmentName)) {
                        jobQueueARN = jobQueueDetail.jobQueueArn();
                    }
                }
            }
            return jobQueueARN;
        });
    }

    /**
     * Disables the specified compute environment asynchronously.
     *
     * @param computeEnvironmentName the name of the compute environment to disable
     * @return a CompletableFuture that completes when the compute environment is
disabled
     */
    public CompletableFuture<UpdateComputeEnvironmentResponse>
disableComputeEnvironmentAsync(String computeEnvironmentName) {
        UpdateComputeEnvironmentRequest updateRequest =
UpdateComputeEnvironmentRequest.builder()
            .computeEnvironment(computeEnvironmentName)
            .state(CEState.DISABLED)
            .build();

        CompletableFuture<UpdateComputeEnvironmentResponse> responseFuture =
getAsyncClient().updateComputeEnvironment(updateRequest);
        responseFuture.whenComplete((response, ex) -> {
            if (ex != null) {
                throw new RuntimeException("Failed to disable compute environment: "
+ ex.getMessage(), ex);
            }
        });

        return responseFuture;
    }

    /**
     * Submits a job asynchronously to the AWS Batch service.
     *
     * @param jobDefinitionName the name of the job definition to use
     * @param jobQueueName the name of the job queue to submit the job to
```

```
    * @param jobARN the Amazon Resource Name (ARN) of the job definition
    * @return a CompletableFuture that, when completed, contains the job ID of the
submitted job
    */
   public CompletableFuture<String> submitJobAsync(String jobDefinitionName, String
jobQueueName, String jobARN) {
       SubmitJobRequest jobRequest = SubmitJobRequest.builder()
           .jobDefinition(jobARN)
           .jobName(jobDefinitionName)
           .jobQueue(jobQueueName)
           .build();

       CompletableFuture<SubmitJobResponse> responseFuture =
getAsyncClient().submitJob(jobRequest);
       responseFuture.whenComplete((response, ex) -> {
           if (ex != null) {
               throw new RuntimeException("Unexpected error occurred: " +
ex.getMessage(), ex);
           }
       });

       return responseFuture.thenApply(SubmitJobResponse::jobId);
   }

   /**
    * Asynchronously retrieves the status of a specific job.
    *
    * @param jobId the ID of the job to retrieve the status for
    * @return a CompletableFuture that completes with the job status
    */
   public CompletableFuture<String> describeJobAsync(String jobId) {
       DescribeJobsRequest describeJobsRequest = DescribeJobsRequest.builder()
           .jobs(jobId)
           .build();

       CompletableFuture<DescribeJobsResponse> responseFuture =
getAsyncClient().describeJobs(describeJobsRequest);
       return responseFuture.whenComplete((response, ex) -> {
           if (ex != null) {
               throw new RuntimeException("Unexpected error occurred: " +
ex.getMessage(), ex);
           }
       }).thenApply(response -> response.jobs().get(0).status().toString());
   }
```

```
   /**
    * Disables the specific job queue using the asynchronous Java client.
    *
    * @param jobQueueArn the Amazon Resource Name (ARN) of the job queue to wait
for
    * @return a {@link CompletableFuture} that completes when the job queue is
disabled
    */
   public CompletableFuture<Void> waitForJobQueueToBeDisabledAsync(String
jobQueueArn) {
       AtomicBoolean isDisabled = new AtomicBoolean(false);
       return CompletableFuture.runAsync(() -> {
           while (!isDisabled.get()) {
               DescribeJobQueuesRequest describeRequest =
DescribeJobQueuesRequest.builder()
                   .jobQueues(jobQueueArn)
                   .build();

               CompletableFuture<DescribeJobQueuesResponse> responseFuture =
getAsyncClient().describeJobQueues(describeRequest);
               responseFuture.whenComplete((describeResponse, ex) -> {
                   if (describeResponse != null) {
                       for (JobQueueDetail jobQueue : describeResponse.jobQueues())
{
                           if (jobQueue.jobQueueArn().equals(jobQueueArn) &&
jobQueue.state() == JQState.DISABLED) {
                               isDisabled.set(true);
                               break;
                           }
                       }
                   } else {
                       throw new RuntimeException("Error describing job queues",
ex);
                   }
               }).join();

               if (!isDisabled.get()) {
                   try {
                       logger.info("Waiting for job queue to be disabled...");
                       Thread.sleep(5000);
                   } catch (InterruptedException e) {
                       Thread.currentThread().interrupt();
```

```
                        throw new RuntimeException("Thread interrupted while waiting
for job queue to be disabled", e);
                    }
                }
            }
        }).whenComplete((result, throwable) -> {
            if (throwable != null) {
                throw new RuntimeException("Error while waiting for job queue to be
disabled", throwable);
            }
        });
    }

    public CompletableFuture<String> getJobQueueARN(String jobQueueName) {
        // Describe the job queue asynchronously
        CompletableFuture<DescribeJobQueuesResponse> describeJobQueuesFuture =
batchClient.describeJobQueues(
            DescribeJobQueuesRequest.builder()
                .jobQueues(jobQueueName)
                .build()
        );

        // Handle the asynchronous response and return the Job Queue ARN in the
CompletableFuture<String>
        CompletableFuture<String> jobQueueArnFuture = new CompletableFuture<>();
        describeJobQueuesFuture.whenComplete((response, error) -> {
            if (error != null) {
                if (error instanceof BatchException) {
                    logger.info("Batch error: " + ((BatchException)
error).awsErrorDetails().errorMessage());
                } else {
                    logger.info("Error describing job queue: " +
error.getMessage());
                }
                jobQueueArnFuture.completeExceptionally(new RuntimeException("Failed
to retrieve Job Queue ARN", error));
            } else {
                if (response.jobQueues().isEmpty()) {
                    jobQueueArnFuture.completeExceptionally(new
RuntimeException("Job queue not found: " + jobQueueName));
                } else {
                    // Assuming only one job queue is returned for the given name
                    String jobQueueArn = response.jobQueues().get(0).jobQueueArn();
                    jobQueueArnFuture.complete(jobQueueArn);
```

```
                }
            }
        });

        return jobQueueArnFuture;
    }

    private static String getComputeEnvironmentName(String computeEnvironment) {
        String[] parts = computeEnvironment.split("/");
        if (parts.length == 2) {
            return parts[1];
        }
        return null;
    }

    public CompletableFuture<String> getAccountId() {
        StsAsyncClient stsAsyncClient = StsAsyncClient.builder()
            .region(Region.US_EAST_1)
            .build();

        return stsAsyncClient.getCallerIdentity()
            .thenApply(GetCallerIdentityResponse::account);
    }


}
```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella Documentazione di riferimento delle API AWS SDK for Java 2.x .
  - CreateComputeEnvironment
  - CreateJobQueue
  - DeleteComputeEnvironment
  - DeleteJobQueue
  - DeregisterJobDefinition
  - DescribeComputeEnvironments
  - DescribeJobQueues
  - DescribeJobs
  - ListJobsPaginator

- RegisterJobDefinition
- SubmitJob
- UpdateComputeEnvironment
- UpdateJobQueue

Azioni

`CreateComputeEnvironment`

Il seguente esempio di codice mostra come usare. `CreateComputeEnvironment`

SDK per Java 2.x

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
   /**
    * Asynchronously creates a new compute environment in AWS Batch.
    *
    * @param computeEnvironmentName the name of the compute environment to create
    * @param batchIAMRole the IAM role to be used by the compute environment
    * @param subnet the subnet ID to be used for the compute environment
    * @param secGroup the security group ID to be used for the compute environment
    * @return a {@link CompletableFuture} representing the asynchronous operation,
which will complete with the
    *         {@link CreateComputeEnvironmentResponse} when the compute environment
has been created
    * @throws BatchException if there is an error creating the compute environment
    * @throws RuntimeException if there is an unexpected error during the operation
    */
   public CompletableFuture<CreateComputeEnvironmentResponse>
createComputeEnvironmentAsync(
       String computeEnvironmentName, String batchIAMRole, String subnet, String
secGroup) {
       CreateComputeEnvironmentRequest environmentRequest =
CreateComputeEnvironmentRequest.builder()
           .computeEnvironmentName(computeEnvironmentName)
```

```
            .type(CEType.MANAGED)
            .state(CEState.ENABLED)
            .computeResources(ComputeResource.builder()
                .type(CRType.FARGATE)
                .maxvCpus(256)
                .subnets(Collections.singletonList(subnet))
                .securityGroupIds(Collections.singletonList(secGroup))
                .build())
            .serviceRole(batchIAMRole)
            .build();

        CompletableFuture<CreateComputeEnvironmentResponse> response =
getAsyncClient().createComputeEnvironment(environmentRequest);
        response.whenComplete((resp, ex) -> {
            if (ex != null) {
               String errorMessage = "Unexpected error occurred: " +
ex.getMessage();
               throw new RuntimeException(errorMessage, ex);
            }
        });

        return response;
    }
```

- Per i dettagli sull'API, consulta la CreateComputeEnvironmentsezione AWS SDK for Java 2.x API Reference.

`CreateJobQueue`

Il seguente esempio di codice mostra come utilizzare`CreateJobQueue`.

SDK per Java 2.x

> **Note**
>
> C'è 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 a job queue asynchronously.
    *
    * @param jobQueueName the name of the job queue to create
    * @param computeEnvironmentName the name of the compute environment to
associate with the job queue
    * @return a CompletableFuture that completes with the Amazon Resource Name
(ARN) of the job queue
    */
   public CompletableFuture<String> createJobQueueAsync(String jobQueueName, String
computeEnvironmentName) {
       if (jobQueueName == null || jobQueueName.isEmpty()) {
           throw new IllegalArgumentException("Job queue name cannot be null or
empty");
       }
       if (computeEnvironmentName == null || computeEnvironmentName.isEmpty()) {
           throw new IllegalArgumentException("Compute environment name cannot be
null or empty");
       }

       CreateJobQueueRequest request = CreateJobQueueRequest.builder()
           .jobQueueName(jobQueueName)
           .priority(1)
           .computeEnvironmentOrder(ComputeEnvironmentOrder.builder()
               .computeEnvironment(computeEnvironmentName)
               .order(1)
               .build())
           .build();

       CompletableFuture<CreateJobQueueResponse> response =
getAsyncClient().createJobQueue(request);
       response.whenComplete((resp, ex) -> {
           if (ex != null) {
               String errorMessage = "Unexpected error occurred: " +
ex.getMessage();
               throw new RuntimeException(errorMessage, ex);
           }
       });

       return response.thenApply(CreateJobQueueResponse::jobQueueArn);
   }
```

- Per i dettagli sull'API, consulta la CreateJobQueuesezione AWS SDK for Java 2.x API Reference.

`DeleteComputeEnvironment`

Il seguente esempio di codice mostra come utilizzare`DeleteComputeEnvironment`.

SDK per Java 2.x

> **Note**
>
> C'è 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 CompletableFuture<DeleteComputeEnvironmentResponse>
deleteComputeEnvironmentAsync(String computeEnvironmentName) {
       DeleteComputeEnvironmentRequest deleteComputeEnvironment =
DeleteComputeEnvironmentRequest.builder()
           .computeEnvironment(computeEnvironmentName)
           .build();

       return getAsyncClient().deleteComputeEnvironment(deleteComputeEnvironment)
           .whenComplete((response, ex) -> {
               if (ex != null) {
                   Throwable cause = ex.getCause();
                   if (cause instanceof BatchException) {
                       throw new RuntimeException(cause);
                   } else {
                       throw new RuntimeException("Unexpected error: " +
cause.getMessage(), cause);
                   }
               }
           });
   }
```

- Per i dettagli sull'API, consulta la DeleteComputeEnvironmentsezione AWS SDK for Java 2.x API Reference.

`DeleteJobQueue`

Il seguente esempio di codice mostra come utilizzare`DeleteJobQueue`.

SDK per Java 2.x

> **Note**
>
> C'è 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 Batch job queue asynchronously.
    *
    * @param jobQueueArn The Amazon Resource Name (ARN) of the job queue to delete.
    * @return A CompletableFuture that represents the asynchronous deletion of the
job queue.
    *         The future completes when the job queue has been successfully deleted
or if an error occurs.
    *         If successful, the future will be completed with a {@code Void}
value.
    *         If an error occurs, the future will be completed exceptionally with
the thrown exception.
    */
   public CompletableFuture<Void> deleteJobQueueAsync(String jobQueueArn) {
       DeleteJobQueueRequest deleteRequest = DeleteJobQueueRequest.builder()
           .jobQueue(jobQueueArn)
           .build();

       CompletableFuture<DeleteJobQueueResponse> responseFuture =
getAsyncClient().deleteJobQueue(deleteRequest);
       return responseFuture.whenComplete((deleteResponse, ex) -> {
           if (ex != null) {
               throw new RuntimeException("Failed to delete job queue: " +
ex.getMessage(), ex);
           }
       }).thenApply(deleteResponse -> null);
   }
```

- Per i dettagli sull'API, consulta la DeleteJobQueuesezione AWS SDK for Java 2.x API Reference.

`DeregisterJobDefinition`

Il seguente esempio di codice mostra come utilizzare`DeregisterJobDefinition`.

SDK per Java 2.x

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
   /**
    * Deregisters a job definition asynchronously.
    *
    * @param jobDefinition the name of the job definition to be deregistered
    * @return a CompletableFuture that completes when the job definition has been
deregistered
    * or an exception has occurred
    */
   public CompletableFuture<DeregisterJobDefinitionResponse>
deregisterJobDefinitionAsync(String jobDefinition) {
       DeregisterJobDefinitionRequest jobDefinitionRequest =
DeregisterJobDefinitionRequest.builder()
           .jobDefinition(jobDefinition)
           .build();

       CompletableFuture<DeregisterJobDefinitionResponse> responseFuture =
getAsyncClient().deregisterJobDefinition(jobDefinitionRequest);
       responseFuture.whenComplete((response, ex) -> {
           if (ex != null) {
               throw new RuntimeException("Unexpected error occurred: " +
ex.getMessage(), ex);
           }
       });

       return responseFuture;
   }
```

- Per i dettagli sull'API, consulta la DeregisterJobDefinitionsezione AWS SDK for Java 2.x API Reference.

`DescribeComputeEnvironments`

Il seguente esempio di codice mostra come utilizzare`DescribeComputeEnvironments`.

SDK per Java 2.x

> **Note**
>
> C'è 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 status of the specified compute environment.
    *
    * @param computeEnvironmentName the name of the compute environment to check
    * @return a CompletableFuture containing the status of the compute environment,
or "ERROR" if an exception occurs
    */
   public CompletableFuture<String> checkComputeEnvironmentsStatus(String
computeEnvironmentName) {
       if (computeEnvironmentName == null || computeEnvironmentName.isEmpty()) {
           throw new IllegalArgumentException("Compute environment name cannot be
null or empty");
       }

       DescribeComputeEnvironmentsRequest environmentsRequest =
DescribeComputeEnvironmentsRequest.builder()
           .computeEnvironments(computeEnvironmentName)
           .build();

       CompletableFuture<DescribeComputeEnvironmentsResponse> response =
getAsyncClient().describeComputeEnvironments(environmentsRequest);
       response.whenComplete((resp, ex) -> {
           if (ex != null) {
```

```
                String errorMessage = "Unexpected error occurred: " +
ex.getMessage();
                throw new RuntimeException(errorMessage, ex);
            }
        });

        return response.thenApply(resp -> resp.computeEnvironments().stream()
            .map(env -> env.statusAsString())
            .findFirst()
            .orElse("UNKNOWN"));
    }
```

- Per i dettagli sull'API, consulta la DescribeComputeEnvironmentssezione AWS SDK for Java 2.x API Reference.

`DescribeJobQueues`

Il seguente esempio di codice mostra come utilizzare`DescribeJobQueues`.

SDK per Java 2.x

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
    /**
     * Asynchronously describes the job queue associated with the specified compute
environment.
     *
     * @param computeEnvironmentName the name of the compute environment to find the
associated job queue for
     * @return a {@link CompletableFuture} that, when completed, contains the job
queue ARN associated with the specified compute environment
     * @throws RuntimeException if the job queue description fails
     */
    public CompletableFuture<String> describeJobQueueAsync(String
computeEnvironmentName) {
```

```
        DescribeJobQueuesRequest describeJobQueuesRequest =
DescribeJobQueuesRequest.builder()
            .build();

        CompletableFuture<DescribeJobQueuesResponse> responseFuture =
getAsyncClient().describeJobQueues(describeJobQueuesRequest);
        return responseFuture.whenComplete((describeJobQueuesResponse, ex) -> {
            if (describeJobQueuesResponse != null) {
                String jobQueueARN;
                for (JobQueueDetail jobQueueDetail :
describeJobQueuesResponse.jobQueues()) {
                    for (ComputeEnvironmentOrder computeEnvironmentOrder :
jobQueueDetail.computeEnvironmentOrder()) {
                        String computeEnvironment =
computeEnvironmentOrder.computeEnvironment();
                        String name = getComputeEnvironmentName(computeEnvironment);
                        if (name.equals(computeEnvironmentName)) {
                            jobQueueARN = jobQueueDetail.jobQueueArn();
                            logger.info("Job queue ARN associated with the compute
environment: " + jobQueueARN);
                        }
                    }
                }
            } else {
                throw new RuntimeException("Failed to describe job queue: " +
ex.getMessage(), ex);
            }
        }).thenApply(describeJobQueuesResponse -> {
            String jobQueueARN = "";
            for (JobQueueDetail jobQueueDetail :
describeJobQueuesResponse.jobQueues()) {
                for (ComputeEnvironmentOrder computeEnvironmentOrder :
jobQueueDetail.computeEnvironmentOrder()) {
                    String computeEnvironment =
computeEnvironmentOrder.computeEnvironment();
                    String name = getComputeEnvironmentName(computeEnvironment);
                    if (name.equals(computeEnvironmentName)) {
                        jobQueueARN = jobQueueDetail.jobQueueArn();
                    }
                }
            }
            return jobQueueARN;
        });
    }
```

- Per i dettagli sull'API, consulta la DescribeJobQueuessezione AWS SDK for Java 2.x API Reference.

`DescribeJobs`

Il seguente esempio di codice mostra come utilizzare`DescribeJobs`.

SDK per Java 2.x

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
   /**
    * Asynchronously retrieves the status of a specific job.
    *
    * @param jobId the ID of the job to retrieve the status for
    * @return a CompletableFuture that completes with the job status
    */
   public CompletableFuture<String> describeJobAsync(String jobId) {
       DescribeJobsRequest describeJobsRequest = DescribeJobsRequest.builder()
           .jobs(jobId)
           .build();

       CompletableFuture<DescribeJobsResponse> responseFuture =
getAsyncClient().describeJobs(describeJobsRequest);
       return responseFuture.whenComplete((response, ex) -> {
           if (ex != null) {
               throw new RuntimeException("Unexpected error occurred: " +
ex.getMessage(), ex);
           }
       }).thenApply(response -> response.jobs().get(0).status().toString());
   }
```

- Per i dettagli sull'API, consulta la DescribeJobssezione AWS SDK for Java 2.x API Reference.

`ListJobsPaginator`

Il seguente esempio di codice mostra come utilizzare`ListJobsPaginator`.

SDK per Java 2.x

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
   /**
    * Asynchronously lists the jobs in the specified job queue with the given job
status.
    *
    * @param jobQueue the name of the job queue to list jobs from
    * @return a List<JobSummary> that contains the jobs that succeeded
    */
   public List<JobSummary> listJobsAsync(String jobQueue) {
       if (jobQueue == null || jobQueue.isEmpty()) {
           throw new IllegalArgumentException("Job queue cannot be null or empty");
       }

       ListJobsRequest listJobsRequest = ListJobsRequest.builder()
           .jobQueue(jobQueue)
           .jobStatus(JobStatus.SUCCEEDED)  // Filter jobs by status.
           .build();

       List<JobSummary> jobSummaries = new ArrayList<>();
       ListJobsPublisher listJobsPaginator =
getAsyncClient().listJobsPaginator(listJobsRequest);
       CompletableFuture<Void> future = listJobsPaginator.subscribe(response -> {
           jobSummaries.addAll(response.jobSummaryList());
       });
       future.join();
       return jobSummaries;
   }
```

- Per i dettagli sull'API, consulta la ListJobsPaginatorsezione AWS SDK for Java 2.x API Reference.

`RegisterJobDefinition`

Il seguente esempio di codice mostra come utilizzare`RegisterJobDefinition`.

SDK per Java 2.x

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
   /**
    * Registers a new job definition asynchronously in AWS Batch.
    * <p>
    * When using Fargate as the compute environment, it is crucial to set the
    * {@link NetworkConfiguration} with {@link AssignPublicIp#ENABLED} to
    * ensure proper networking configuration for the Fargate tasks. This
    * allows the tasks to communicate with external services, access the
    * internet, or communicate within a VPC.
    *
    * @param jobDefinitionName the name of the job definition to be registered
    * @param executionRoleARN the ARN (Amazon Resource Name) of the execution role
    *                         that provides permissions for the containers in the
job
    * @param cpuArch a value of either X86_64 or ARM64 required for the service
call
    * @return a CompletableFuture that completes with the ARN of the registered
    *         job definition upon successful execution, or completes exceptionally
with
    *         an error if the registration fails
    */
   public CompletableFuture<String> registerJobDefinitionAsync(String
jobDefinitionName, String executionRoleARN, String image, String cpuArch) {
       NetworkConfiguration networkConfiguration = NetworkConfiguration.builder()
               .assignPublicIp(AssignPublicIp.ENABLED)
               .build();

       ContainerProperties containerProperties = ContainerProperties.builder()
               .image(image)
               .executionRoleArn(executionRoleARN)
               .resourceRequirements(
                       Arrays.asList(
```

```
                                ResourceRequirement.builder()
                                        .type(ResourceType.VCPU)
                                        .value("1")
                                        .build(),
                                ResourceRequirement.builder()
                                        .type(ResourceType.MEMORY)
                                        .value("2048")
                                        .build()
                        )
                )
                .networkConfiguration(networkConfiguration)
               .runtimePlatform(b -> b
                        .cpuArchitecture(cpuArch)
                        .operatingSystemFamily("LINUX"))
                .build();

        RegisterJobDefinitionRequest request =
RegisterJobDefinitionRequest.builder()
                .jobDefinitionName(jobDefinitionName)
                .type(JobDefinitionType.CONTAINER)
                .containerProperties(containerProperties)
                .platformCapabilities(PlatformCapability.FARGATE)
                .build();

        CompletableFuture<String> future = new CompletableFuture<>();
        getAsyncClient().registerJobDefinition(request)
                .thenApply(RegisterJobDefinitionResponse::jobDefinitionArn)
                .whenComplete((result, ex) -> {
                    if (ex != null) {
                        future.completeExceptionally(ex);
                    } else {
                        future.complete(result);
                    }
                });

        return future;
    }
```

- Per i dettagli sull'API, consulta la RegisterJobDefinitionsezione AWS SDK for Java 2.x API Reference.

`SubmitJob`

Il seguente esempio di codice mostra come utilizzare`SubmitJob`.

SDK per Java 2.x

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
   /**
    * Submits a job asynchronously to the AWS Batch service.
    *
    * @param jobDefinitionName the name of the job definition to use
    * @param jobQueueName the name of the job queue to submit the job to
    * @param jobARN the Amazon Resource Name (ARN) of the job definition
    * @return a CompletableFuture that, when completed, contains the job ID of the
submitted job
    */
   public CompletableFuture<String> submitJobAsync(String jobDefinitionName, String
jobQueueName, String jobARN) {
       SubmitJobRequest jobRequest = SubmitJobRequest.builder()
           .jobDefinition(jobARN)
           .jobName(jobDefinitionName)
           .jobQueue(jobQueueName)
           .build();

       CompletableFuture<SubmitJobResponse> responseFuture =
getAsyncClient().submitJob(jobRequest);
       responseFuture.whenComplete((response, ex) -> {
           if (ex != null) {
               throw new RuntimeException("Unexpected error occurred: " +
ex.getMessage(), ex);
           }
       });

       return responseFuture.thenApply(SubmitJobResponse::jobId);
   }
```

- Per i dettagli sull'API, consulta la SubmitJobsezione AWS SDK for Java 2.x API Reference.

`UpdateComputeEnvironment`

Il seguente esempio di codice mostra come utilizzare`UpdateComputeEnvironment`.

SDK per Java 2.x

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
   /**
    * Disables the specified compute environment asynchronously.
    *
    * @param computeEnvironmentName the name of the compute environment to disable
    * @return a CompletableFuture that completes when the compute environment is
disabled
    */
   public CompletableFuture<UpdateComputeEnvironmentResponse>
disableComputeEnvironmentAsync(String computeEnvironmentName) {
       UpdateComputeEnvironmentRequest updateRequest =
UpdateComputeEnvironmentRequest.builder()
           .computeEnvironment(computeEnvironmentName)
           .state(CEState.DISABLED)
           .build();

       CompletableFuture<UpdateComputeEnvironmentResponse> responseFuture =
getAsyncClient().updateComputeEnvironment(updateRequest);
       responseFuture.whenComplete((response, ex) -> {
           if (ex != null) {
               throw new RuntimeException("Failed to disable compute environment: "
+ ex.getMessage(), ex);
           }
       });

       return responseFuture;
   }
```

- Per i dettagli sull'API, consulta la UpdateComputeEnvironmentsezione AWS SDK for Java 2.x API Reference.

`UpdateJobQueue`

Il seguente esempio di codice mostra come utilizzare`UpdateJobQueue`.

SDK per Java 2.x

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
   /**
    * Disables the specified job queue asynchronously.
    *
    * @param jobQueueArn the Amazon Resource Name (ARN) of the job queue to be
disabled
    * @return a {@link CompletableFuture} that completes when the job queue update
operation is complete,
    *         or completes exceptionally if an error occurs during the operation
    */
   public CompletableFuture<Void> disableJobQueueAsync(String jobQueueArn) {
       UpdateJobQueueRequest updateRequest = UpdateJobQueueRequest.builder()
           .jobQueue(jobQueueArn)
           .state(JQState.DISABLED)
           .build();

       CompletableFuture<UpdateJobQueueResponse> responseFuture =
getAsyncClient().updateJobQueue(updateRequest);
       return responseFuture.whenComplete((updateResponse, ex) -> {
           if (ex != null) {
               throw new RuntimeException("Failed to update job queue: " +
ex.getMessage(), ex);
           }
       }).thenApply(updateResponse -> null);
   }
```

- Per i dettagli sull'API, consulta la UpdateJobQueuesezione AWS SDK for Java 2.x API Reference.

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.

Ogni esempio include un collegamento al codice sorgente completo, 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 GetFoundationModella 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. ListFoundationModelsAWS 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.

Gli scenari sono esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio o combinate con altri Servizi AWS.

Ogni esempio include un collegamento al codice sorgente completo, dove puoi trovare istruzioni su come configurare ed eseguire il codice nel contesto.

Argomenti

- Scenari
- AI21 Laboratori Jurassic-2

- Amazon Nova
- Amazon Nova Tela
- Amazon Titan Image Generator
- Testo Amazon Titan
- Amazon Titan Text Embeddings
- Anthropic Claude
- Cohere Command
- Meta Llama
- IA Mistral
- Diffusione stabile

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

Genera video da istruzioni di testo utilizzando Amazon Bedrock

Il seguente esempio di codice mostra come creare un'app Spring Boot che genera video da istruzioni di testo utilizzando Amazon Bedrock e il modello Nova-Reel.

SDK per Java 2.x

> **Note**
>
> GitHubC'è di più su. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

Genera video da istruzioni di testo utilizzando Amazon Bedrock e Nova-Reel.

```
import org.springframework.stereotype.Service;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.core.document.Document;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
import software.amazon.awssdk.services.bedrockruntime.model.*;

import java.util.concurrent.CompletableFuture;

@Service
public class VideoGenerationService {

    public GenerateVideoResponse generateVideo(String prompt) {

        // add S3 bucket you want to store your generated videos
        String s3Bucket = "s3://mygeneratedvidoenovatest";


        //Create json request as an instance of Document class
        Document novaRequest = prepareDocument(prompt);

        // Create request
        StartAsyncInvokeRequest request = StartAsyncInvokeRequest.builder()
                .modelId("amazon.nova-reel-v1:0")
                .modelInput(novaRequest)
                .outputDataConfig(AsyncInvokeOutputDataConfig.builder()

 .s3OutputDataConfig(AsyncInvokeS3OutputDataConfig.builder().s3Uri(s3Bucket).build())
```

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

        try (BedrockRuntimeAsyncClient bedrockClient =
getBedrockRuntimeAsyncClient()) {
            CompletableFuture<StartAsyncInvokeResponse>
startAsyncInvokeResponseCompletableFuture =
bedrockClient.startAsyncInvoke(request);

            //blocking operation to wait for the AWS API response
            StartAsyncInvokeResponse startAsyncInvokeResponse =
startAsyncInvokeResponseCompletableFuture.get();
            System.out.println("invocation ARN: " +
startAsyncInvokeResponse.invocationArn());

            GenerateVideoResponse response = new GenerateVideoResponse();
            response.setStatus("inProgress");
            response.setExecutionArn(startAsyncInvokeResponse.invocationArn());

            return response;
        } catch (Exception e) {
            System.out.println(e);
            throw new RuntimeException(e);
        }

    }

    public GenerateVideoResponse checkGenerationStatus(String invocationArn) {
        GenerateVideoResponse response = new GenerateVideoResponse();

        try (BedrockRuntimeAsyncClient bedrockClient =
getBedrockRuntimeAsyncClient()) {
            //creating async request to fetch status by invocation Arn
            GetAsyncInvokeRequest asyncRequest =
GetAsyncInvokeRequest.builder().invocationArn(invocationArn).build();

            CompletableFuture<GetAsyncInvokeResponse> asyncInvoke =
bedrockClient.getAsyncInvoke(asyncRequest);

            //blocking operation to wait for the AWS API response
            GetAsyncInvokeResponse asyncInvokeResponse = asyncInvoke.get();
            System.out.println("Invocation status =" +
asyncInvokeResponse.statusAsString());
```

```
            response.setExecutionArn(invocationArn);
            response.setStatus(asyncInvokeResponse.statusAsString());
            return response;
        } catch (Exception e) {
            e.printStackTrace();
            throw new RuntimeException(e);
        }

    }

    private static BedrockRuntimeAsyncClient getBedrockRuntimeAsyncClient() {
        BedrockRuntimeAsyncClient bedrockClient =
 BedrockRuntimeAsyncClient.builder()
                .region(Region.US_EAST_1)
                .credentialsProvider(ProfileCredentialsProvider.create())
                .build();
        return bedrockClient;
    }

    private static Document prepareDocument(String prompt) {
        Document textToVideoParams = Document.mapBuilder()
                .putString("text", prompt)
                .build();

        Document videoGenerationConfig = Document.mapBuilder()
                .putNumber("durationSeconds", 6)
                .putNumber("fps", 24)
                .putString("dimension", "1280x720")
                .build();

        Document novaRequest = Document.mapBuilder()
                .putString("taskType", "TEXT_VIDEO")
                .putDocument("textToVideoParams", textToVideoParams)
                .putDocument("videoGenerationConfig", videoGenerationConfig)
                .build();
        return novaRequest;
    }
}
```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella Documentazione di riferimento delle API AWS SDK for Java 2.x .
  - GetAsyncInvoke

- StartAsyncInvoke

Utilizzo dello strumento con l'API Converse

Il seguente esempio di codice mostra come creare un'interazione tipica tra un'applicazione, un modello di intelligenza artificiale generativa e strumenti connessi o come APIs mediare le interazioni tra l'IA e il mondo esterno. Utilizza l'esempio del collegamento di un'API meteorologica esterna al modello di intelligenza artificiale in modo che possa fornire informazioni meteorologiche in tempo reale basate sull'input dell'utente.

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.

L'esecuzione principale del flusso dello scenario. Questo scenario orchestra la conversazione tra l'utente, l'API Amazon Bedrock Converse e uno strumento meteorologico.

```
/*
 This demo illustrates a tool use scenario using Amazon Bedrock's Converse API and a
 weather tool.
 The program interacts with a foundation model on Amazon Bedrock to provide weather
 information based on user
 input. It uses the Open-Meteo API (https://open-meteo.com) to retrieve current
 weather data for a given location.
 */
public class BedrockScenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");
    private static String modelId = "amazon.nova-lite-v1:0";
    private static String defaultPrompt = "What is the weather like in Seattle?";
    private static WeatherTool weatherTool = new WeatherTool();

    // The maximum number of recursive calls allowed in the tool use function.
    // This helps prevent infinite loops and potential performance issues.
    private static int maxRecursions = 5;
    static BedrockActions bedrockActions = new BedrockActions();
    public static boolean interactive = true;
```

```
    private static final String systemPrompt = """
            You are a weather assistant that provides current weather data for user-
specified locations using only
            the Weather_Tool, which expects latitude and longitude. Infer the
 coordinates from the location yourself.
            If the user provides coordinates, infer the approximate location and
 refer to it in your response.
            To use the tool, you strictly apply the provided tool specification.

            - Explain your step-by-step process, and give brief updates before each
 step.
            - Only use the Weather_Tool for data. Never guess or make up
 information.
            - Repeat the tool use for subsequent requests if necessary.
            - If the tool errors, apologize, explain weather is unavailable, and
 suggest other options.
            - Report temperatures in °C (°F) and wind in km/h (mph). Keep weather
 reports concise. Sparingly use
              emojis where appropriate.
            - Only respond to weather queries. Remind off-topic users of your
 purpose.
            - Never claim to search online, access external data, or use tools
 besides Weather_Tool.
            - Complete the entire process until you have all required data before
 sending the complete response.
            """;

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("""
                ==================================================
                Welcome to the Amazon Bedrock Tool Use demo!
                ==================================================

                This assistant provides current weather information for user-
specified locations.
                You can ask for weather details by providing the location name or
 coordinates.

                Example queries:
                - What's the weather like in New York?
                - Current weather for latitude 40.70, longitude -74.01
                - Is it warmer in Rome or Barcelona today?
```

```
                To exit the program, simply type 'x' and press Enter.

                P.S.: You're not limited to single locations, or even to using
English!
                Have fun and experiment with the app!
                """);
        System.out.println(DASHES);

        try {
            runConversation(scanner);

        } catch (Exception ex) {
            System.out.println("There was a problem running the scenario: " +
ex.getMessage());
        }

        waitForInputToContinue(scanner);

        System.out.println(DASHES);
        System.out.println("Amazon Bedrock Converse API with Tool Use Feature
Scenario is complete.");
        System.out.println(DASHES);
    }

    /**
     * Starts the conversation with the user and handles the interaction with
Bedrock.
     */
    private static List<Message> runConversation(Scanner scanner) {
        List<Message> conversation = new ArrayList<>();

        // Get the first user input
        String userInput = getUserInput("Your weather info request:", scanner);
        System.out.println(userInput);

        while (userInput != null) {
            ContentBlock block = ContentBlock.builder()
                    .text(userInput)
                    .build();

            List<ContentBlock> blockList = new ArrayList<>();
            blockList.add(block);

            Message message = Message.builder()
```

```
                    .role(ConversationRole.USER)
                    .content(blockList)
                    .build();

            conversation.add(message);

            // Send the conversation to Amazon Bedrock.
            ConverseResponse bedrockResponse =
sendConversationToBedrock(conversation);

            // Recursively handle the model's response until the model has returned
its final response or the recursion counter has reached 0.
            processModelResponse(bedrockResponse, conversation, maxRecursions);

            // Repeat the loop until the user decides to exit the application.
            userInput = getUserInput("Your weather info request:", scanner);
        }
        printFooter();
        return conversation;
    }

    /**
     * Processes the response from the model and updates the conversation
accordingly.
     *
     * @param modelResponse the response from the model
     * @param conversation  the ongoing conversation
     * @param maxRecursion  the maximum number of recursions allowed
     */
    private static void processModelResponse(ConverseResponse modelResponse,
List<Message> conversation, int maxRecursion) {
        if (maxRecursion <= 0) {
            // Stop the process, the number of recursive calls could indicate an
infinite loop
            System.out.println("\tWarning: Maximum number of recursions reached.
Please try again.");
        }

        // Append the model's response to the ongoing conversation
        conversation.add(modelResponse.output().message());

        String modelResponseVal = modelResponse.stopReasonAsString();
        if (modelResponseVal.compareTo("tool_use") == 0) {
```

```
            // If the stop reason is "tool_use", forward everything to the tool use
handler
            handleToolUse(modelResponse.output(), conversation, maxRecursion - 1);
        }

        if (modelResponseVal.compareTo("end_turn") == 0) {
            // If the stop reason is "end_turn", print the model's response text,
and finish the process

PrintModelResponse(modelResponse.output().message().content().get(0).text());
            if (!interactive) {
                defaultPrompt = "x";
            }
        }
    }

    /**
     * Handles the use of a tool by the model in a conversation.
     *
     * @param modelResponse the response from the model, which may include a tool
use request
     * @param conversation  the current conversation, which will be updated with the
tool use results
     * @param maxRecursion  the maximum number of recursive calls allowed to handle
the model's response
     */
    private static void handleToolUse(ConverseOutput modelResponse, List<Message>
conversation, int maxRecursion) {
        List<ContentBlock> toolResults = new ArrayList<>();

        // The model's response can consist of multiple content blocks
        for (ContentBlock contentBlock : modelResponse.message().content()) {
            if (contentBlock.text() != null && !contentBlock.text().isEmpty()) {
                // If the content block contains text, print it to the console
                PrintModelResponse(contentBlock.text());
            }

            if (contentBlock.toolUse() != null) {
                ToolResponse toolResponse = invokeTool(contentBlock.toolUse());

                // Add the tool use ID and the tool's response to the list of
results
                List<ToolResultContentBlock> contentBlockList = new ArrayList<>();
                ToolResultContentBlock block = ToolResultContentBlock.builder()
```

```
                        .json(toolResponse.getContent())
                        .build();
                contentBlockList.add(block);

                ToolResultBlock toolResultBlock = ToolResultBlock.builder()
                        .toolUseId(toolResponse.getToolUseId())
                        .content(contentBlockList)
                        .build();

                ContentBlock contentBlock1 = ContentBlock.builder()
                        .toolResult(toolResultBlock)
                        .build();

                toolResults.add(contentBlock1);
            }
        }

        // Embed the tool results in a new user message
        Message message = Message.builder()
                .role(ConversationRole.USER)
                .content(toolResults)
                .build();

        // Append the new message to the ongoing conversation
        //conversation.add(message);
        conversation.add(message);

        // Send the conversation to Amazon Bedrock
        var response = sendConversationToBedrock(conversation);

        // Recursively handle the model's response until the model has returned its
final response or the recursion counter has reached 0
        processModelResponse(response, conversation, maxRecursion);
    }

    // Invokes the specified tool with the given payload and returns the tool's
response.
    // If the requested tool does not exist, an error message is returned.
    private static ToolResponse invokeTool(ToolUseBlock payload) {
        String toolName = payload.name();

        if (Objects.equals(toolName, "Weather_Tool")) {
            Map<String, Document> inputData = payload.input().asMap();
            printToolUse(toolName, inputData);
```

```
            // Invoke the weather tool with the input data provided
            Document weatherResponse =
weatherTool.fetchWeatherData(inputData.get("latitude").toString(),
inputData.get("longitude").toString());

            ToolResponse toolResponse = new ToolResponse();
            toolResponse.setContent(weatherResponse);
            toolResponse.setToolUseId(payload.toolUseId());
            return toolResponse;
        } else {
            String errorMessage = "The requested tool with name " + toolName + "
does not exist.";
            System.out.println(errorMessage);
            return null;
        }
    }

    public static void printToolUse(String toolName, Map<String, Document>
inputData) {
        System.out.println("Invoking tool: " + toolName + " with input: " +
inputData.get("latitude").toString() + ", " + inputData.get("longitude").toString()
+ "...");
    }

    private static void PrintModelResponse(String message) {
        System.out.println("\tThe model's response:\n");
        System.out.println(message);
        System.out.println("");
    }

    private static ConverseResponse sendConversationToBedrock(List<Message>
conversation) {
        System.out.println("Calling Bedrock...");

        try {
            return bedrockActions.sendConverseRequestAsync(modelId, systemPrompt,
conversation, weatherTool.getToolSpec());
        } catch (ModelNotReadyException ex) {
             System.err.println("Model is not ready. Please try again later: " +
ex.getMessage());
            throw ex;
        } catch (BedrockRuntimeException ex) {
            System.err.println("Bedrock service error: " + ex.getMessage());
```

```
            throw ex;
        } catch (RuntimeException ex) {
            System.err.println("Unexpected error occurred: " + ex.getMessage());
            throw ex;
        }
    }

    private static ConverseResponse sendConversationToBedrockwithSpec(List<Message>
conversation, ToolSpecification toolSpec) {
        System.out.println("Calling Bedrock...");

        // Send the conversation, system prompt, and tool configuration, and return
the response
        return bedrockActions.sendConverseRequestAsync(modelId, systemPrompt,
conversation, toolSpec);
    }

    public static String getUserInput(String prompt, Scanner scanner) {
        String userInput = defaultPrompt;
        if (interactive) {
            System.out.println("*".repeat(80));
            System.out.println(prompt + " (x to exit): \n\t");
            userInput = scanner.nextLine();
        }

        if (userInput == null || userInput.trim().isEmpty()) {
            return getUserInput("\tPlease enter your weather info request, e.g., the
name of a city", scanner);
        }

        if (userInput.equalsIgnoreCase("x")) {
            return null;
        }

        return userInput;
    }

    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.");
            }
        }
    }

    public static void printFooter() {
        System.out.println("""
                ================================================
                Thank you for checking out the Amazon Bedrock Tool Use demo. We hope
 you
                learned something new, or got some inspiration for your own apps
 today!

                For more Bedrock examples in different programming languages, have a
 look at:
                https://docs.aws.amazon.com/bedrock/latest/userguide/
service_code_examples.html
                ================================================
                """);
    }
}
```

Lo strumento meteorologico utilizzato dalla demo. Questo file definisce le specifiche dello strumento e implementa la logica per recuperare i dati meteorologici utilizzando l'API Open-Meteo.

```
public class WeatherTool {

    private static final Logger logger = LoggerFactory.getLogger(WeatherTool.class);
    private static java.net.http.HttpClient httpClient = null;

    /**
     * Returns the JSON Schema specification for the Weather tool. The tool
 specification
     * defines the input schema and describes the tool's functionality.
     * For more information, see https://json-schema.org/understanding-json-schema/
reference.
```

```
     *
     * @return The tool specification for the Weather tool.
     */
    public ToolSpecification getToolSpec() {
        Map<String, Document> latitudeMap = new HashMap<>();
        latitudeMap.put("type", Document.fromString("string"));
        latitudeMap.put("description", Document.fromString("Geographical WGS84
latitude of the location."));

        // Create the nested "longitude" object
        Map<String, Document> longitudeMap = new HashMap<>();
        longitudeMap.put("type", Document.fromString("string"));
        longitudeMap.put("description", Document.fromString("Geographical WGS84
longitude of the location."));

        // Create the "properties" object
        Map<String, Document> propertiesMap = new HashMap<>();
        propertiesMap.put("latitude", Document.fromMap(latitudeMap));
        propertiesMap.put("longitude", Document.fromMap(longitudeMap));

        // Create the "required" array
        List<Document> requiredList = new ArrayList<>();
        requiredList.add(Document.fromString("latitude"));
        requiredList.add(Document.fromString("longitude"));

        // Create the root object
        Map<String, Document> rootMap = new HashMap<>();
        rootMap.put("type", Document.fromString("object"));
        rootMap.put("properties", Document.fromMap(propertiesMap));
        rootMap.put("required", Document.fromList(requiredList));

        // Now create the Document representing the JSON schema
        Document document = Document.fromMap(rootMap);

        ToolSpecification specification = ToolSpecification.builder()
            .name("Weather_Tool")
            .description("Get the current weather for a given location, based on its
WGS84 coordinates.")
            .inputSchema(ToolInputSchema.builder()
                .json(document)
                .build())
            .build();

        return specification;
```

```
    }

    /**
     * Fetches weather data for the given latitude and longitude.
     *
     * @param latitude  the latitude coordinate
     * @param longitude the longitude coordinate
     * @return a {@link CompletableFuture} containing the weather data as a JSON
 string
     */
    public Document fetchWeatherData(String latitude, String longitude) {
        HttpClient httpClient = HttpClient.newHttpClient();

        // Ensure no extra double quotes
        latitude = latitude.replace("\"", "");
        longitude = longitude.replace("\"", "");

        String endpoint = "https://api.open-meteo.com/v1/forecast";
        String url = String.format("%s?latitude=%s&longitude=
%s&current_weather=True", endpoint, latitude, longitude);

        HttpRequest request = HttpRequest.newBuilder()
            .uri(URI.create(url))
            .build();

        try {
            HttpResponse<String> response = httpClient.send(request,
 HttpResponse.BodyHandlers.ofString());
            if (response.statusCode() == 200) {
                String weatherJson = response.body();
                System.out.println(weatherJson);
                ObjectMapper objectMapper = new ObjectMapper();
                Map<String, Object> rawMap = objectMapper.readValue(weatherJson, new
 TypeReference<Map<String, Object>>() {});
                Map<String, Document> documentMap = convertToDocumentMap(rawMap);


                Document weatherDocument = Document.fromMap(documentMap);
                System.out.println(weatherDocument);
                return weatherDocument;
            } else {
                throw new RuntimeException("Error fetching weather data: " +
 response.statusCode());
            }
```

```
        } catch (Exception e) {
            System.out.println("Error fetching weather data: " + e.getMessage());
            throw new RuntimeException("Error fetching weather data", e);
        }

    }

    private static Map<String, Document> convertToDocumentMap(Map<String, Object>
 inputMap) {
        Map<String, Document> result = new HashMap<>();
        for (Map.Entry<String, Object> entry : inputMap.entrySet()) {
            result.put(entry.getKey(), convertToDocument(entry.getValue()));
        }
        return result;
    }

    // Convert different types of Objects to Document
    private static Document convertToDocument(Object value) {
        if (value instanceof Map) {
            return Document.fromMap(convertToDocumentMap((Map<String, Object>)
 value));
        } else if (value instanceof Integer) {
            return Document.fromNumber(SdkNumber.fromInteger((Integer) value));
        } else if (value instanceof Double) {  //
            return Document.fromNumber(SdkNumber.fromDouble((Double) value));
        } else if (value instanceof Boolean) {
            return Document.fromBoolean((Boolean) value);
        } else if (value instanceof String) {
            return Document.fromString((String) value);
        }
        return Document.fromNull(); // Handle null values safely
    }
}
```

L'azione dell'API Converse con una configurazione dello strumento.

```
    /**
     * Sends an asynchronous converse request to the AI model.
     *
     * @param modelId      the unique identifier of the AI model to be used for the
 converse request
     * @param systemPrompt the system prompt to be included in the converse request
```

```
    * @param conversation a list of messages representing the conversation history
    * @param toolSpec     the specification of the tool to be used in the converse
request
    * @return the converse response received from the AI model
    */
   public ConverseResponse sendConverseRequestAsync(String modelId, String
systemPrompt, List<Message> conversation, ToolSpecification toolSpec) {
       List<Tool> toolList = new ArrayList<>();
       Tool tool = Tool.builder()
           .toolSpec(toolSpec)
           .build();

       toolList.add(tool);

       ToolConfiguration configuration = ToolConfiguration.builder()
           .tools(toolList)
           .build();

       SystemContentBlock block = SystemContentBlock.builder()
           .text(systemPrompt)
           .build();

       ConverseRequest request = ConverseRequest.builder()
           .modelId(modelId)
           .system(block)
           .messages(conversation)
           .toolConfig(configuration)
           .build();

       try {
           ConverseResponse response = getClient().converse(request).join();
           return response;

       } catch (ModelNotReadyException ex) {
           throw new RuntimeException("Model is not ready: " + ex.getMessage(),
ex);
       } catch (BedrockRuntimeException ex) {
           throw new RuntimeException("Failed to converse with Bedrock model: " +
ex.getMessage(), ex);
       }
   }
```

- Per i dettagli sull'API, consulta Converse in AWS SDK for Java 2.x API Reference.

AI21 Laboratori 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 InvokeModelin AWS SDK for Java 2.x API Reference.

Amazon Nova

conversare

Il seguente esempio di codice mostra come inviare un messaggio di testo ad Amazon Nova, 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 Nova utilizzando l'API Converse di Bedrock con il client Java asincrono.

```
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.*;

import java.util.concurrent.CompletableFuture;

/**
 * This example demonstrates how to use the Amazon Nova foundation models
 * with an asynchronous Amazon Bedrock runtime client to generate text.
 * It shows how to:
 * - Set up the Amazon Bedrock runtime client
 * - Create a message
 * - Configure and send a request
 * - Process the response
 */
public class ConverseAsync {

    public static String converseAsync() {

        // Step 1: Create the Amazon Bedrock runtime client
        // The runtime client handles the communication with AI models on Amazon
 Bedrock
```

```
        BedrockRuntimeAsyncClient client = BedrockRuntimeAsyncClient.builder()
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_EAST_1)
                .build();

        // Step 2: Specify which model to use
        // Available Amazon Nova models and their characteristics:
        // - Amazon Nova Micro: Text-only model optimized for lowest latency and
 cost
        // - Amazon Nova Lite:  Fast, low-cost multimodal model for image, video,
 and text
        // - Amazon Nova Pro:   Advanced multimodal model balancing accuracy, speed,
 and cost
        //
        // For the latest available models, see:
        // https://docs.aws.amazon.com/bedrock/latest/userguide/models-
supported.html
        String modelId = "amazon.nova-lite-v1:0";

        // Step 3: Create the message
        // The message includes the text prompt and specifies that it comes from the
 user
        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();

        // Step 4: Configure the request
        // Optional parameters to control the model's response:
        // - maxTokens: maximum number of tokens to generate
        // - temperature: randomness (max: 1.0, default: 0.7)
        //   OR
        // - topP: diversity of word choice (max: 1.0, default: 0.9)
        // Note: Use either temperature OR topP, but not both
        ConverseRequest request = ConverseRequest.builder()
                .modelId(modelId)
                .messages(message)
                .inferenceConfig(config -> config
                                .maxTokens(500)     // The maximum response length
                                .temperature(0.5F)  // Using temperature for
 randomness control
```

```
                        //.topP(0.9F)       // Alternative: use topP instead of
 temperature
                ).build();

        // Step 5: Send and process the request asynchronously
        // - Send the request to the model
        // - Extract and return the generated text from the response
        try {
            CompletableFuture<ConverseResponse> asyncResponse =
 client.converse(request);
            return asyncResponse.thenApply(
                    response -> response.output().message().content().get(0).text()
            ).get();

        } catch (Exception e) {
            System.err.printf("Can't invoke '%s': %s", modelId, e.getMessage());
            throw new RuntimeException(e);
        }
    }

    public static void main(String[] args) {
        String response = converseAsync();
        System.out.println(response);
    }
}
```

Invia un messaggio di testo ad Amazon Nova utilizzando l'API Converse di Bedrock.

```
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.*;

/**
 * This example demonstrates how to use the Amazon Nova foundation models
 * with a synchronous Amazon Bedrock runtime client to generate text.
 * It shows how to:
 * - Set up the Amazon Bedrock runtime client
 * - Create a message
```

```
 * - Configure and send a request
 * - Process the response
 */
public class Converse {

    public static String converse() {

        // Step 1: Create the Amazon Bedrock runtime client
        // The runtime client handles the communication with AI models on Amazon
 Bedrock
        BedrockRuntimeClient client = BedrockRuntimeClient.builder()
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_EAST_1)
                .build();

        // Step 2: Specify which model to use
        // Available Amazon Nova models and their characteristics:
        // - Amazon Nova Micro: Text-only model optimized for lowest latency and
 cost
        // - Amazon Nova Lite:  Fast, low-cost multimodal model for image, video,
 and text
        // - Amazon Nova Pro:   Advanced multimodal model balancing accuracy, speed,
 and cost
        //
        // For the latest available models, see:
        // https://docs.aws.amazon.com/bedrock/latest/userguide/models-
supported.html
        String modelId = "amazon.nova-lite-v1:0";

        // Step 3: Create the message
        // The message includes the text prompt and specifies that it comes from the
 user
        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();

        // Step 4: Configure the request
        // Optional parameters to control the model's response:
        // - maxTokens: maximum number of tokens to generate
        // - temperature: randomness (max: 1.0, default: 0.7)
        //   OR
```

```
        // - topP: diversity of word choice (max: 1.0, default: 0.9)
        // Note: Use either temperature OR topP, but not both
        ConverseRequest request = ConverseRequest.builder()
                .modelId(modelId)
                .messages(message)
                .inferenceConfig(config -> config
                                .maxTokens(500)     // The maximum response length
                                .temperature(0.5F)  // Using temperature for
 randomness control
                        //.topP(0.9F)       // Alternative: use topP instead of
 temperature
                ).build();

        // Step 5: Send and process the request
        // - Send the request to the model
        // - Extract and return the generated text from the response
        try {
            ConverseResponse response = client.converse(request);
            return response.output().message().content().get(0).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) {
        String response = converse();
        System.out.println(response);
    }
}
```

- Per i dettagli sull'API, consulta Converse in AWS SDK for Java 2.x API Reference.

ConverseStream

Il seguente esempio di codice mostra come inviare un messaggio di testo ad Amazon Nova, 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 Nova utilizzando l'API Converse di Bedrock ed elabora il flusso di risposta in tempo reale.

```
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.*;

import java.util.concurrent.ExecutionException;

/**
 * This example demonstrates how to use the Amazon Nova foundation models with an
 * asynchronous Amazon Bedrock runtime client to generate streaming text responses.
 * It shows how to:
 * - Set up the Amazon Bedrock runtime client
 * - Create a message
 * - Configure a streaming request
 * - Set up a stream handler to process the response chunks
 * - Process the streaming response
 */
public class ConverseStream {

    public static void converseStream() {

        // Step 1: Create the Amazon Bedrock runtime client
        // The runtime client handles the communication with AI models on Amazon
 Bedrock
        BedrockRuntimeAsyncClient client = BedrockRuntimeAsyncClient.builder()
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_EAST_1)
                .build();

        // Step 2: Specify which model to use
```

```
        // Available Amazon Nova models and their characteristics:
        // - Amazon Nova Micro: Text-only model optimized for lowest latency and
 cost
        // - Amazon Nova Lite:  Fast, low-cost multimodal model for image, video,
 and text
        // - Amazon Nova Pro:   Advanced multimodal model balancing accuracy, speed,
 and cost
        //
        // For the latest available models, see:
        // https://docs.aws.amazon.com/bedrock/latest/userguide/models-
supported.html
        String modelId = "amazon.nova-lite-v1:0";

        // Step 3: Create the message
        // The message includes the text prompt and specifies that it comes from the
 user
        var inputText = "Describe the purpose of a 'hello world' program in one
 paragraph";
        var message = Message.builder()
                .content(ContentBlock.fromText(inputText))
                .role(ConversationRole.USER)
                .build();

        // Step 4: Configure the request
        // Optional parameters to control the model's response:
        // - maxTokens: maximum number of tokens to generate
        // - temperature: randomness (max: 1.0, default: 0.7)
        //   OR
        // - topP: diversity of word choice (max: 1.0, default: 0.9)
        // Note: Use either temperature OR topP, but not both
        ConverseStreamRequest request = ConverseStreamRequest.builder()
                .modelId(modelId)
                .messages(message)
                .inferenceConfig(config -> config
                                .maxTokens(500)     // The maximum response length
                                .temperature(0.5F)  // Using temperature for
 randomness control
                        //.topP(0.9F)       // Alternative: use topP instead of
 temperature
                ).build();

        // Step 5: Set up the stream handler
        // The stream handler processes chunks of the response as they arrive
        // - onContentBlockDelta: Processes each text chunk
```

```
        // - onError: Handles any errors during streaming
        var streamHandler = ConverseStreamResponseHandler.builder()
                .subscriber(ConverseStreamResponseHandler.Visitor.builder()
                        .onContentBlockDelta(chunk -> {
                            System.out.print(chunk.delta().text());
                            System.out.flush();  // Ensure immediate output of each
 chunk
                        }).build())
                .onError(err -> System.err.printf("Can't invoke '%s': %s", modelId,
 err.getMessage()))
                .build();

        // Step 6: Send the streaming request and process the response
        // - Send the request to the model
        // - Attach the handler to process response chunks as they arrive
        // - Handle any errors during streaming
        try {
            client.converseStream(request, streamHandler).get();

        } catch (ExecutionException | InterruptedException e) {
            System.err.printf("Can't invoke '%s': %s", modelId,
 e.getCause().getMessage());
        }
    }

    public static void main(String[] args) {
        converseStream();
    }
}
```

- Per i dettagli sull'API, consulta la sezione AWS SDK for Java 2.x API ConverseStreamReference.

Scenario: utilizzo dello strumento con l'API Converse

Il seguente esempio di codice mostra come creare un'interazione tipica tra un'applicazione, un modello di intelligenza artificiale generativa e strumenti connessi o come APIs mediare le interazioni tra l'IA e il mondo esterno. Utilizza l'esempio del collegamento di un'API meteorologica esterna al modello di intelligenza artificiale in modo che possa fornire informazioni meteorologiche in tempo reale basate sull'input dell'utente.

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.

L'esecuzione principale del flusso dello scenario. Questo scenario orchestra la conversazione tra l'utente, l'API Amazon Bedrock Converse e uno strumento meteorologico.

```
/*
 This demo illustrates a tool use scenario using Amazon Bedrock's Converse API and a
 weather tool.
 The program interacts with a foundation model on Amazon Bedrock to provide weather
 information based on user
 input. It uses the Open-Meteo API (https://open-meteo.com) to retrieve current
 weather data for a given location.
 */
public class BedrockScenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");
    private static String modelId = "amazon.nova-lite-v1:0";
    private static String defaultPrompt = "What is the weather like in Seattle?";
    private static WeatherTool weatherTool = new WeatherTool();

    // The maximum number of recursive calls allowed in the tool use function.
    // This helps prevent infinite loops and potential performance issues.
    private static int maxRecursions = 5;
    static BedrockActions bedrockActions = new BedrockActions();
    public static boolean interactive = true;

    private static final String systemPrompt = """
            You are a weather assistant that provides current weather data for user-
specified locations using only
            the Weather_Tool, which expects latitude and longitude. Infer the
 coordinates from the location yourself.
            If the user provides coordinates, infer the approximate location and
 refer to it in your response.
            To use the tool, you strictly apply the provided tool specification.

            - Explain your step-by-step process, and give brief updates before each
 step.
```

```
            - Only use the Weather_Tool for data. Never guess or make up
 information.
            - Repeat the tool use for subsequent requests if necessary.
            - If the tool errors, apologize, explain weather is unavailable, and
 suggest other options.
            - Report temperatures in °C (°F) and wind in km/h (mph). Keep weather
 reports concise. Sparingly use
              emojis where appropriate.
            - Only respond to weather queries. Remind off-topic users of your
 purpose.
            - Never claim to search online, access external data, or use tools
 besides Weather_Tool.
            - Complete the entire process until you have all required data before
 sending the complete response.
            """;

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("""
                ==================================================
                Welcome to the Amazon Bedrock Tool Use demo!
                ==================================================

                This assistant provides current weather information for user-
specified locations.
                You can ask for weather details by providing the location name or
 coordinates.

                Example queries:
                - What's the weather like in New York?
                - Current weather for latitude 40.70, longitude -74.01
                - Is it warmer in Rome or Barcelona today?

                To exit the program, simply type 'x' and press Enter.

                P.S.: You're not limited to single locations, or even to using
 English!
                Have fun and experiment with the app!
                """);
        System.out.println(DASHES);

        try {
            runConversation(scanner);
```

```
        } catch (Exception ex) {
            System.out.println("There was a problem running the scenario: " +
ex.getMessage());
        }

        waitForInputToContinue(scanner);

        System.out.println(DASHES);
        System.out.println("Amazon Bedrock Converse API with Tool Use Feature
Scenario is complete.");
        System.out.println(DASHES);
    }

    /**
     * Starts the conversation with the user and handles the interaction with
Bedrock.
     */
    private static List<Message> runConversation(Scanner scanner) {
        List<Message> conversation = new ArrayList<>();

        // Get the first user input
        String userInput = getUserInput("Your weather info request:", scanner);
        System.out.println(userInput);

        while (userInput != null) {
            ContentBlock block = ContentBlock.builder()
                    .text(userInput)
                    .build();

            List<ContentBlock> blockList = new ArrayList<>();
            blockList.add(block);

            Message message = Message.builder()
                    .role(ConversationRole.USER)
                    .content(blockList)
                    .build();

            conversation.add(message);

            // Send the conversation to Amazon Bedrock.
            ConverseResponse bedrockResponse =
sendConversationToBedrock(conversation);
```

```
            // Recursively handle the model's response until the model has returned
its final response or the recursion counter has reached 0.
            processModelResponse(bedrockResponse, conversation, maxRecursions);

            // Repeat the loop until the user decides to exit the application.
            userInput = getUserInput("Your weather info request:", scanner);
        }
        printFooter();
        return conversation;
    }

    /**
     * Processes the response from the model and updates the conversation
accordingly.
     *
     * @param modelResponse the response from the model
     * @param conversation  the ongoing conversation
     * @param maxRecursion  the maximum number of recursions allowed
     */
    private static void processModelResponse(ConverseResponse modelResponse,
List<Message> conversation, int maxRecursion) {
        if (maxRecursion <= 0) {
            // Stop the process, the number of recursive calls could indicate an
infinite loop
            System.out.println("\tWarning: Maximum number of recursions reached.
Please try again.");
        }

        // Append the model's response to the ongoing conversation
        conversation.add(modelResponse.output().message());

        String modelResponseVal = modelResponse.stopReasonAsString();
        if (modelResponseVal.compareTo("tool_use") == 0) {
            // If the stop reason is "tool_use", forward everything to the tool use
handler
            handleToolUse(modelResponse.output(), conversation, maxRecursion - 1);
        }

        if (modelResponseVal.compareTo("end_turn") == 0) {
            // If the stop reason is "end_turn", print the model's response text,
and finish the process

PrintModelResponse(modelResponse.output().message().content().get(0).text());
            if (!interactive) {
```

```
                defaultPrompt = "x";
            }
        }
    }

    /**
     * Handles the use of a tool by the model in a conversation.
     *
     * @param modelResponse the response from the model, which may include a tool
use request
     * @param conversation  the current conversation, which will be updated with the
tool use results
     * @param maxRecursion  the maximum number of recursive calls allowed to handle
the model's response
     */
    private static void handleToolUse(ConverseOutput modelResponse, List<Message>
conversation, int maxRecursion) {
        List<ContentBlock> toolResults = new ArrayList<>();

        // The model's response can consist of multiple content blocks
        for (ContentBlock contentBlock : modelResponse.message().content()) {
            if (contentBlock.text() != null && !contentBlock.text().isEmpty()) {
                // If the content block contains text, print it to the console
                PrintModelResponse(contentBlock.text());
            }

            if (contentBlock.toolUse() != null) {
                ToolResponse toolResponse = invokeTool(contentBlock.toolUse());

                // Add the tool use ID and the tool's response to the list of
results
                List<ToolResultContentBlock> contentBlockList = new ArrayList<>();
                ToolResultContentBlock block = ToolResultContentBlock.builder()
                        .json(toolResponse.getContent())
                        .build();
                contentBlockList.add(block);

                ToolResultBlock toolResultBlock = ToolResultBlock.builder()
                        .toolUseId(toolResponse.getToolUseId())
                        .content(contentBlockList)
                        .build();

                ContentBlock contentBlock1 = ContentBlock.builder()
                        .toolResult(toolResultBlock)
```

```
                        .build();

                toolResults.add(contentBlock1);
            }
        }

        // Embed the tool results in a new user message
        Message message = Message.builder()
                .role(ConversationRole.USER)
                .content(toolResults)
                .build();

        // Append the new message to the ongoing conversation
        //conversation.add(message);
        conversation.add(message);

        // Send the conversation to Amazon Bedrock
        var response = sendConversationToBedrock(conversation);

        // Recursively handle the model's response until the model has returned its
final response or the recursion counter has reached 0
        processModelResponse(response, conversation, maxRecursion);
    }

    // Invokes the specified tool with the given payload and returns the tool's
response.
    // If the requested tool does not exist, an error message is returned.
    private static ToolResponse invokeTool(ToolUseBlock payload) {
        String toolName = payload.name();

        if (Objects.equals(toolName, "Weather_Tool")) {
            Map<String, Document> inputData = payload.input().asMap();
            printToolUse(toolName, inputData);

            // Invoke the weather tool with the input data provided
            Document weatherResponse =
weatherTool.fetchWeatherData(inputData.get("latitude").toString(),
inputData.get("longitude").toString());

            ToolResponse toolResponse = new ToolResponse();
            toolResponse.setContent(weatherResponse);
            toolResponse.setToolUseId(payload.toolUseId());
            return toolResponse;
        } else {
```

```
            String errorMessage = "The requested tool with name " + toolName + "
does not exist.";
            System.out.println(errorMessage);
            return null;
        }
    }

    public static void printToolUse(String toolName, Map<String, Document>
inputData) {
        System.out.println("Invoking tool: " + toolName + " with input: " +
inputData.get("latitude").toString() + ", " + inputData.get("longitude").toString()
+ "...");
    }

    private static void PrintModelResponse(String message) {
        System.out.println("\tThe model's response:\n");
        System.out.println(message);
        System.out.println("");
    }

    private static ConverseResponse sendConversationToBedrock(List<Message>
conversation) {
        System.out.println("Calling Bedrock...");

        try {
            return bedrockActions.sendConverseRequestAsync(modelId, systemPrompt,
conversation, weatherTool.getToolSpec());
        } catch (ModelNotReadyException ex) {
             System.err.println("Model is not ready. Please try again later: " +
ex.getMessage());
            throw ex;
        } catch (BedrockRuntimeException ex) {
            System.err.println("Bedrock service error: " + ex.getMessage());
            throw ex;
        } catch (RuntimeException ex) {
            System.err.println("Unexpected error occurred: " + ex.getMessage());
            throw ex;
        }
    }

    private static ConverseResponse sendConversationToBedrockwithSpec(List<Message>
conversation, ToolSpecification toolSpec) {
        System.out.println("Calling Bedrock...");
```

```
        // Send the conversation, system prompt, and tool configuration, and return
the response
        return bedrockActions.sendConverseRequestAsync(modelId, systemPrompt,
conversation, toolSpec);
    }

    public static String getUserInput(String prompt, Scanner scanner) {
        String userInput = defaultPrompt;
        if (interactive) {
            System.out.println("*".repeat(80));
            System.out.println(prompt + " (x to exit): \n\t");
            userInput = scanner.nextLine();
        }

        if (userInput == null || userInput.trim().isEmpty()) {
            return getUserInput("\tPlease enter your weather info request, e.g., the
name of a city", scanner);
        }

        if (userInput.equalsIgnoreCase("x")) {
            return null;
        }

        return userInput;
    }

    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.");
            }
        }
    }

    public static void printFooter() {
```

```
        System.out.println("""
                ==================================================
                Thank you for checking out the Amazon Bedrock Tool Use demo. We hope
 you
                learned something new, or got some inspiration for your own apps
 today!

                For more Bedrock examples in different programming languages, have a
 look at:
                https://docs.aws.amazon.com/bedrock/latest/userguide/
service_code_examples.html
                ==================================================
                """);
    }
}
```

Lo strumento meteorologico utilizzato dalla demo. Questo file definisce le specifiche dello strumento e implementa la logica per recuperare i dati meteorologici utilizzando l'API Open-Meteo.

```
public class WeatherTool {

    private static final Logger logger = LoggerFactory.getLogger(WeatherTool.class);
    private static java.net.http.HttpClient httpClient = null;

    /**
     * Returns the JSON Schema specification for the Weather tool. The tool
 specification
     * defines the input schema and describes the tool's functionality.
     * For more information, see https://json-schema.org/understanding-json-schema/
reference.
     *
     * @return The tool specification for the Weather tool.
     */
    public ToolSpecification getToolSpec() {
        Map<String, Document> latitudeMap = new HashMap<>();
        latitudeMap.put("type", Document.fromString("string"));
        latitudeMap.put("description", Document.fromString("Geographical WGS84
 latitude of the location."));

        // Create the nested "longitude" object
        Map<String, Document> longitudeMap = new HashMap<>();
```

```
        longitudeMap.put("type", Document.fromString("string"));
        longitudeMap.put("description", Document.fromString("Geographical WGS84
longitude of the location."));

        // Create the "properties" object
        Map<String, Document> propertiesMap = new HashMap<>();
        propertiesMap.put("latitude", Document.fromMap(latitudeMap));
        propertiesMap.put("longitude", Document.fromMap(longitudeMap));

        // Create the "required" array
        List<Document> requiredList = new ArrayList<>();
        requiredList.add(Document.fromString("latitude"));
        requiredList.add(Document.fromString("longitude"));

        // Create the root object
        Map<String, Document> rootMap = new HashMap<>();
        rootMap.put("type", Document.fromString("object"));
        rootMap.put("properties", Document.fromMap(propertiesMap));
        rootMap.put("required", Document.fromList(requiredList));

        // Now create the Document representing the JSON schema
        Document document = Document.fromMap(rootMap);

        ToolSpecification specification = ToolSpecification.builder()
            .name("Weather_Tool")
            .description("Get the current weather for a given location, based on its
WGS84 coordinates.")
            .inputSchema(ToolInputSchema.builder()
                .json(document)
                .build())
            .build();

        return specification;
    }

    /**
     * Fetches weather data for the given latitude and longitude.
     *
     * @param latitude  the latitude coordinate
     * @param longitude the longitude coordinate
     * @return a {@link CompletableFuture} containing the weather data as a JSON
string
     */
    public Document fetchWeatherData(String latitude, String longitude) {
```

```
        HttpClient httpClient = HttpClient.newHttpClient();

        // Ensure no extra double quotes
        latitude = latitude.replace("\"", "");
        longitude = longitude.replace("\"", "");

        String endpoint = "https://api.open-meteo.com/v1/forecast";
        String url = String.format("%s?latitude=%s&longitude=
%s&current_weather=True", endpoint, latitude, longitude);

        HttpRequest request = HttpRequest.newBuilder()
            .uri(URI.create(url))
            .build();

        try {
            HttpResponse<String> response = httpClient.send(request,
 HttpResponse.BodyHandlers.ofString());
            if (response.statusCode() == 200) {
                String weatherJson = response.body();
                System.out.println(weatherJson);
                ObjectMapper objectMapper = new ObjectMapper();
                Map<String, Object> rawMap = objectMapper.readValue(weatherJson, new
 TypeReference<Map<String, Object>>() {});
                Map<String, Document> documentMap = convertToDocumentMap(rawMap);


                Document weatherDocument = Document.fromMap(documentMap);
                System.out.println(weatherDocument);
                return weatherDocument;
            } else {
                throw new RuntimeException("Error fetching weather data: " +
 response.statusCode());
            }
        } catch (Exception e) {
            System.out.println("Error fetching weather data: " + e.getMessage());
            throw new RuntimeException("Error fetching weather data", e);
        }

    }

    private static Map<String, Document> convertToDocumentMap(Map<String, Object>
 inputMap) {
        Map<String, Document> result = new HashMap<>();
        for (Map.Entry<String, Object> entry : inputMap.entrySet()) {
```

```
            result.put(entry.getKey(), convertToDocument(entry.getValue()));
        }
        return result;
    }

    // Convert different types of Objects to Document
    private static Document convertToDocument(Object value) {
        if (value instanceof Map) {
            return Document.fromMap(convertToDocumentMap((Map<String, Object>)
 value));
        } else if (value instanceof Integer) {
            return Document.fromNumber(SdkNumber.fromInteger((Integer) value));
        } else if (value instanceof Double) {  //
            return Document.fromNumber(SdkNumber.fromDouble((Double) value));
        } else if (value instanceof Boolean) {
            return Document.fromBoolean((Boolean) value);
        } else if (value instanceof String) {
            return Document.fromString((String) value);
        }
        return Document.fromNull(); // Handle null values safely
    }
}
```

L'azione dell'API Converse con una configurazione dello strumento.

```
    /**
     * Sends an asynchronous converse request to the AI model.
     *
     * @param modelId      the unique identifier of the AI model to be used for the
 converse request
     * @param systemPrompt the system prompt to be included in the converse request
     * @param conversation a list of messages representing the conversation history
     * @param toolSpec     the specification of the tool to be used in the converse
 request
     * @return the converse response received from the AI model
     */
    public ConverseResponse sendConverseRequestAsync(String modelId, String
 systemPrompt, List<Message> conversation, ToolSpecification toolSpec) {
        List<Tool> toolList = new ArrayList<>();
        Tool tool = Tool.builder()
            .toolSpec(toolSpec)
            .build();
```

```
        toolList.add(tool);

        ToolConfiguration configuration = ToolConfiguration.builder()
            .tools(toolList)
            .build();

        SystemContentBlock block = SystemContentBlock.builder()
            .text(systemPrompt)
            .build();

        ConverseRequest request = ConverseRequest.builder()
            .modelId(modelId)
            .system(block)
            .messages(conversation)
            .toolConfig(configuration)
            .build();

        try {
            ConverseResponse response = getClient().converse(request).join();
            return response;

        } catch (ModelNotReadyException ex) {
            throw new RuntimeException("Model is not ready: " + ex.getMessage(),
ex);
        } catch (BedrockRuntimeException ex) {
            throw new RuntimeException("Failed to converse with Bedrock model: " +
ex.getMessage(), ex);
        }
    }
```

- Per i dettagli sull'API, consulta Converse in AWS SDK for Java 2.x API Reference.

Amazon Nova Tela

InvokeModel

Il seguente esempio di codice mostra come richiamare Amazon Nova Canvas 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 Nova Canvas.

```
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 software.amazon.awssdk.services.bedrockruntime.model.InvokeModelResponse;

import java.security.SecureRandom;
import java.util.Base64;

import static com.example.bedrockruntime.libs.ImageTools.displayImage;

/**
 * This example demonstrates how to use Amazon Nova Canvas to generate images.
 * It shows how to:
 * - Set up the Amazon Bedrock runtime client
 * - Configure the image generation parameters
 * - Send a request to generate an image
 * - Process the response and handle the generated image
 */
public class InvokeModel {

    public static byte[] invokeModel() {

        // Step 1: Create the Amazon Bedrock runtime client
        // The runtime client handles the communication with AI models on Amazon
 Bedrock
        BedrockRuntimeClient client = BedrockRuntimeClient.builder()
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_EAST_1)
```

```
                .build();

        // Step 2: Specify which model to use
        // For the latest available models, see:
        // https://docs.aws.amazon.com/bedrock/latest/userguide/models-
supported.html
        String modelId = "amazon.nova-canvas-v1:0";

        // Step 3: Configure the generation parameters and create the request
        // First, set the main parameters:
        // - prompt: Text description of the image to generate
        // - seed: Random number for reproducible generation (0 to 858,993,459)
        String prompt = "A stylized picture of a cute old steampunk robot";
        int seed = new SecureRandom().nextInt(858_993_460);

        // Then, create the request using a template with the following structure:
        // - taskType: TEXT_IMAGE (specifies text-to-image generation)
        // - textToImageParams: Contains the text prompt
        // - imageGenerationConfig: Contains optional generation settings (seed,
 quality, etc.)
        // For a list of available request parameters, see:
        // https://docs.aws.amazon.com/nova/latest/userguide/image-gen-req-resp-
structure.html
        String request = """
                {
                    "taskType": "TEXT_IMAGE",
                    "textToImageParams": {
                        "text": "{{prompt}}"
                    },
                    "imageGenerationConfig": {
                        "seed": {{seed}},
                        "quality": "standard"
                    }
                }"""
                .replace("{{prompt}}", prompt)
                .replace("{{seed}}", String.valueOf(seed));

        // Step 4: Send and process the request
        // - Send the request to the model using InvokeModelResponse
        // - Extract the Base64-encoded image from the JSON response
        // - Convert the encoded image to a byte array and return it
        try {
            InvokeModelResponse response = client.invokeModel(builder -> builder
                    .modelId(modelId)
```

```
                    .body(SdkBytes.fromUtf8String(request))
            );

            JSONObject responseBody = new
 JSONObject(response.body().asUtf8String());
            // Convert the Base64 string to byte array for better handling
            return Base64.getDecoder().decode(
                    new JSONPointer("/images/0").queryFrom(responseBody).toString()
            );

        } catch (SdkClientException e) {
            System.err.printf("ERROR: Can't invoke '%s'. Reason: %s%n", modelId,
 e.getMessage());
            throw new RuntimeException(e);
        }
    }

    public static void main(String[] args) {
        System.out.println("Generating image. This may take a few seconds...");
        byte[] imageData = invokeModel();
        displayImage(imageData);
    }
}
```

- Per i dettagli sull'API, consulta la InvokeModelsezione 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 JSONPointer("/
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.xAPI 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 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.

```
// 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 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 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 la sezione API ConverseStreamReference 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 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 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 InvokeModelin 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.InvokeModelWithResponseStreamResponseH

import java.util.concurrent.ExecutionException;

import static
 software.amazon.awssdk.services.bedrockruntime.model.InvokeModelWithResponseStreamResponseH

public class InvokeModelWithResponseStream {

    public static String invokeModelWithResponseStream() {

        // 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.xAPI 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. InvokeModelAWS 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 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.

```
// 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().getFirst().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().getFirst().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 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 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 sulle API, consulta ConverseStreamla sezione 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 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 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 InvokeModelin 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.InvokeModelWithResponseStreamResponseH

import java.util.Objects;
import java.util.concurrent.ExecutionException;

import static
 software.amazon.awssdk.services.bedrockruntime.model.InvokeModelWithResponseStreamResponseH

public class InvokeModelWithResponseStream {

    public static String invokeModelWithResponseStream() {

        // 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.xAPI Reference.

Ragionamento

Il seguente esempio di codice mostra come utilizzare la funzionalità di ragionamento di Anthropic Claude 3.7 Sonnet su Amazon 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.

Usa la funzionalità di ragionamento di Anthropic Claude 3.7 Sonnet con il client di runtime asincrono Bedrock.

```
import com.example.bedrockruntime.models.anthropicClaude.lib.ReasoningResponse;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.document.Document;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
import software.amazon.awssdk.services.bedrockruntime.model.*;

import java.util.concurrent.CompletableFuture;

/**
 * This example demonstrates how to use Anthropic Claude 3.7 Sonnet's reasoning
 capability
```

```
 * with an asynchronous Amazon Bedrock runtime client.
 * It shows how to:
 * - Set up the Amazon Bedrock async runtime client
 * - Create a message
 * - Configure reasoning parameters
 * - Send an asynchronous request with reasoning enabled
 * - Process both the reasoning output and final response
 */
public class ReasoningAsync {

    public static ReasoningResponse reasoningAsync() {

        // Create the Amazon Bedrock runtime client
        var client = BedrockRuntimeAsyncClient.builder()
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_EAST_1)
                .build();

        // Specify the model ID. For the latest available models, see:
        // https://docs.aws.amazon.com/bedrock/latest/userguide/models-
supported.html
        var modelId = "us.anthropic.claude-3-7-sonnet-20250219-v1:0";

        // Create the message with the user's prompt
        var prompt = "Describe the purpose of a 'hello world' program in one line.";
        var message = Message.builder()
                .content(ContentBlock.fromText(prompt))
                .role(ConversationRole.USER)
                .build();

        // Configure reasoning parameters with a 2000 token budget
        Document reasoningConfig = Document.mapBuilder()
                .putDocument("thinking", Document.mapBuilder()
                        .putString("type", "enabled")
                        .putNumber("budget_tokens", 2000)
                        .build())
                .build();

        try {
            // Send message and reasoning configuration to the model
            CompletableFuture<ConverseResponse> asyncResponse =
 client.converse(request -> request
                    .additionalModelRequestFields(reasoningConfig)
                    .messages(message)
```

```
                    .modelId(modelId)
            );

            // Process the response asynchronously
            return asyncResponse.thenApply(response -> {

                        var content = response.output().message().content();
                        ReasoningContentBlock reasoning = null;
                        String text = null;

                        // Process each content block to find reasoning and response
 text
                        for (ContentBlock block : content) {
                            if (block.reasoningContent() != null) {
                                reasoning = block.reasoningContent();
                            } else if (block.text() != null) {
                                text = block.text();
                            }
                        }

                        return new ReasoningResponse(reasoning, text);
                    }
            ).get();

        } catch (Exception e) {
            System.err.printf("Can't invoke '%s': %s", modelId, e.getMessage());
            throw new RuntimeException(e);
        }
    }

    public static void main(String[] args) {
        // Execute the example and display reasoning and final response
        ReasoningResponse response = reasoningAsync();
        System.out.println("\n<thinking>");
        System.out.println(response.reasoning().reasoningText());
        System.out.println("</thinking>\n");
        System.out.println(response.text());
    }
}
```

Usa la funzionalità di ragionamento di Anthropic Claude 3.7 Sonnet con il client di runtime sincrono Bedrock.

```
import com.example.bedrockruntime.models.anthropicClaude.lib.ReasoningResponse;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.document.Document;
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.*;

/**
 * This example demonstrates how to use Anthropic Claude 3.7 Sonnet's reasoning
 capability
 * with the synchronous Amazon Bedrock runtime client.
 * It shows how to:
 * - Set up the Amazon Bedrock runtime client
 * - Create a message
 * - Configure reasoning parameters
 * - Send a request with reasoning enabled
 * - Process both the reasoning output and final response
 */
public class Reasoning {

    public static ReasoningResponse reasoning() {

        // Create the Amazon Bedrock runtime client
        var client = BedrockRuntimeClient.builder()
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_EAST_1)
                .build();

        // Specify the model ID. For the latest available models, see:
        // https://docs.aws.amazon.com/bedrock/latest/userguide/models-
supported.html
        var modelId = "us.anthropic.claude-3-7-sonnet-20250219-v1:0";

        // Create the message with the user's prompt
        var prompt = "Describe the purpose of a 'hello world' program in one line.";
        var message = Message.builder()
                .content(ContentBlock.fromText(prompt))
                .role(ConversationRole.USER)
                .build();

        // Configure reasoning parameters with a 2000 token budget
```

```
        Document reasoningConfig = Document.mapBuilder()
                .putDocument("thinking", Document.mapBuilder()
                        .putString("type", "enabled")
                        .putNumber("budget_tokens", 2000)
                        .build())
                .build();

        try {
            // Send message and reasoning configuration to the model
            ConverseResponse bedrockResponse = client.converse(request -> request
                    .additionalModelRequestFields(reasoningConfig)
                    .messages(message)
                    .modelId(modelId)
            );


            // Extract both reasoning and final response
            var content = bedrockResponse.output().message().content();
            ReasoningContentBlock reasoning = null;
            String text = null;

            // Process each content block to find reasoning and response text
            for (ContentBlock block : content) {
                if (block.reasoningContent() != null) {
                    reasoning = block.reasoningContent();
                } else if (block.text() != null) {
                    text = block.text();
                }
            }

            return new ReasoningResponse(reasoning, 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) {
        // Execute the example and display reasoning and final response
        ReasoningResponse response = reasoning();
        System.out.println("\n<thinking>");
        System.out.println(response.reasoning().reasoningText());
```

```
        System.out.println("</thinking>\n");
        System.out.println(response.text());
    }
}
```

- Per i dettagli sulle API, consulta Converse in API Reference.AWS SDK for Java 2.x

Ragionamento con una risposta in streaming

Il seguente esempio di codice mostra come utilizzare la funzionalità di ragionamento di Anthropic Claude 3.7 Sonnet su Amazon 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.

Usa la funzionalità di ragionamento di Anthropic Claude 3.7 Sonnet per generare risposte di testo in streaming.

```
import com.example.bedrockruntime.models.anthropicClaude.lib.ReasoningResponse;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.document.Document;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
import software.amazon.awssdk.services.bedrockruntime.model.*;

import java.util.concurrent.ExecutionException;
import java.util.concurrent.atomic.AtomicReference;

/**
 * This example demonstrates how to use Anthropic Claude 3.7 Sonnet's reasoning
 * capability to generate streaming text responses.
 * It shows how to:
 * - Set up the Amazon Bedrock runtime client
 * - Create a message
```

```
 * - Configure a streaming request
 * - Set up a stream handler to process the response chunks
 * - Process the streaming response
 */
public class ReasoningStream {

    public static ReasoningResponse reasoningStream() {

        // Create the Amazon Bedrock runtime client
        var client = BedrockRuntimeAsyncClient.builder()
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_EAST_1)
                .build();

        // Specify the model ID. For the latest available models, see:
        // https://docs.aws.amazon.com/bedrock/latest/userguide/models-
supported.html
        var modelId = "us.anthropic.claude-3-7-sonnet-20250219-v1:0";

        // Create the message with the user's prompt
        var prompt = "Describe the purpose of a 'hello world' program in one line.";
        var message = Message.builder()
                .content(ContentBlock.fromText(prompt))
                .role(ConversationRole.USER)
                .build();

        // Configure reasoning parameters with a 2000 token budget
        Document reasoningConfig = Document.mapBuilder()
                .putDocument("thinking", Document.mapBuilder()
                        .putString("type", "enabled")
                        .putNumber("budget_tokens", 2000)
                        .build())
                .build();

        // Configure the request with the message, model ID, and reasoning config
        ConverseStreamRequest request = ConverseStreamRequest.builder()
                .additionalModelRequestFields(reasoningConfig)
                .messages(message)
                .modelId(modelId)
                .build();

        StringBuilder reasoning = new StringBuilder();
        StringBuilder text = new StringBuilder();
        AtomicReference<ReasoningResponse> finalresponse = new AtomicReference<>();
```

```
        // Set up the stream handler to processes chunks of the response as they
arrive
        var streamHandler = ConverseStreamResponseHandler.builder()
                .subscriber(ConverseStreamResponseHandler.Visitor.builder()
                        .onContentBlockDelta(chunk -> {
                            ContentBlockDelta delta = chunk.delta();
                            if (delta.reasoningContent() != null) {
                                if (reasoning.isEmpty()) {
                                    System.out.println("\n<thinking>");
                                }
                                if (delta.reasoningContent().text() != null) {

System.out.print(delta.reasoningContent().text());

reasoning.append(delta.reasoningContent().text());
                                }
                            } else if (delta.text() != null) {
                                if (text.isEmpty()) {
                                    System.out.println("\n</thinking>\n");
                                }
                                System.out.print(delta.text());
                                text.append(delta.text());
                            }
                            System.out.flush();  // Ensure immediate output of each
chunk
                        }).build())
                .onComplete(() -> finalresponse.set(new ReasoningResponse(
                        ReasoningContentBlock.fromReasoningText(t ->
t.text(reasoning.toString())),
                        text.toString()
                )))
                .onError(err -> System.err.printf("Can't invoke '%s': %s", modelId,
err.getMessage()))
                .build();

        // Step 6: Send the streaming request and process the response
        // - Send the request to the model
        // - Attach the handler to process response chunks as they arrive
        // - Handle any errors during streaming
        try {
            client.converseStream(request, streamHandler).get();
            return finalresponse.get();
```

```
        } catch (ExecutionException | InterruptedException e) {
            System.err.printf("Can't invoke '%s': %s", modelId,
 e.getCause().getMessage());
            throw new RuntimeException(e);
        } catch (Exception e) {
            System.err.printf("Can't invoke '%s': %s", modelId, e.getMessage());
            throw new RuntimeException(e);
        }
    }

    public static void main(String[] args) {
        reasoningStream();
    }
}
```

- Per i dettagli sulle API, consulta Converse in API Reference.AWS SDK for Java 2.x

Cohere Command

conversare

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

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 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 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 la sezione API ConverseStreamReference 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 InvokeModelin 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 InvokeModelin 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.InvokeModelWithResponseStreamResponseH

import java.util.concurrent.ExecutionException;

import static
 software.amazon.awssdk.services.bedrockruntime.model.InvokeModelWithResponseStreamResponseH

public class Command_R_InvokeModelWithResponseStream {

    public static String invokeModelWithResponseStream() {

        // 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.xAPI 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.InvokeModelWithResponseStreamResponseH

import java.util.concurrent.ExecutionException;

import static
 software.amazon.awssdk.services.bedrockruntime.model.InvokeModelWithResponseStreamResponseH

public class Command_InvokeModelWithResponseStream {

    public static String invokeModelWithResponseStream() {

        // 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.xAPI Reference.

Meta Llama

conversare

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

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 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 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 la sezione API ConverseStreamReference AWS SDK for Java 2.x .

InvokeModel

Il seguente esempio di codice mostra come inviare un messaggio di testo a Meta Llama, 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 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_WEST_2)
                .build();

        // Set the model ID, e.g., Llama 3 70b Instruct.
        var modelId = "meta.llama3-70b-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|><|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 InvokeModelin AWS SDK for Java 2.x API Reference.

InvokeModelWithResponseStream

Il seguente esempio di codice mostra come inviare un messaggio di testo a Meta Llama, 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.InvokeModelWithResponseStreamResponseH

import java.util.concurrent.ExecutionException;

import static
 software.amazon.awssdk.services.bedrockruntime.model.InvokeModelWithResponseStreamResponseH

public class Llama3_InvokeModelWithResponseStream {

    public static String invokeModelWithResponseStream() {

        // 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_WEST_2)
                .build();
```

```
        // Set the model ID, e.g., Llama 3 70b Instruct.
        var modelId = "meta.llama3-70b-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|><|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.xAPI 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 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 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 ConverseStreamAPI 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 InvokeModelin 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.InvokeModelWithResponseStreamResponseH

import java.util.concurrent.ExecutionException;

import static
 software.amazon.awssdk.services.bedrockruntime.model.InvokeModelWithResponseStreamResponseH

public class InvokeModelWithResponseStream {

    public static String invokeModelWithResponseStream() {
```

```
        // 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.xAPI Reference.

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.xAPI 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.

Gli scenari sono esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio o combinate con altri Servizi AWS.

Ogni esempio include un collegamento al codice sorgente completo, in cui 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 CloudFrontWaiterdi 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))
                                                                .trustedKeyGroups(b3
 -> b3

 .quantity(1)

 .items(keyGroupId)

 .enabled(true))
                                                                .allowedMethods(b4 -
> 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 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 CreateFunctionsezione 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 i cookie firmati URLs o registrati.

```
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 CreateKeyGroupsezione AWS SDK for Java 2.x API 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 DeleteDistributionAPI 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 UpdateDistributionsezione 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 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 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

Segno URLs e cookie

Il seguente esempio di codice mostra come creare cookie firmati URLs e cookie 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 CannedSignerRequestclasse per firmare URLs 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 CustomSignerRequestclasse per firmare URLs 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)
                // .resourceUrlPattern("https://*.example.com/*")  // Optional.
                .privateKey(path)
                .keyPairId(publicKeyId)
                .expirationDate(expireDate)
                .activeDate(activeDate) // Optional.
                // .ipRange("192.168.0.1/24") // Optional.
                .build();
    }
}
```

L'esempio seguente mostra l'uso della CloudFrontUtilitiesclasse per produrre cookie firmati e URLs. 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 CloudFrontUtilitiessezione 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 nozioni di base sono esempi di codice che mostrano come eseguire le operazioni essenziali all'interno di un servizio.

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.

Gli scenari sono esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio o combinate con altri Servizi AWS.

Ogni esempio include un collegamento al codice sorgente completo, in cui è possibile trovare istruzioni su come configurare ed eseguire il codice nel contesto.

Nozioni di base

Salve CloudWatch

L'esempio di codice seguente mostra come iniziare a utilizzare CloudWatch.

SDK per Java 2.x

> **Note**
>
> C'è 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.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 ListMetricssezione AWS SDK for Java 2.x API Reference.

Argomenti

- Nozioni di base
- Azioni
- Scenari

Nozioni di base

Informazioni di base

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.

Esegui uno scenario interattivo che dimostri le CloudWatch funzionalità.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import
 software.amazon.awssdk.services.cloudwatch.model.DashboardInvalidInputErrorException;
import software.amazon.awssdk.services.cloudwatch.model.DeleteAlarmsResponse;
import
 software.amazon.awssdk.services.cloudwatch.model.DeleteAnomalyDetectorResponse;
import software.amazon.awssdk.services.cloudwatch.model.DeleteDashboardsResponse;
import software.amazon.awssdk.services.cloudwatch.model.Dimension;
import software.amazon.awssdk.services.cloudwatch.model.GetMetricStatisticsResponse;
import software.amazon.awssdk.services.cloudwatch.model.LimitExceededException;
import software.amazon.awssdk.services.cloudwatch.model.PutDashboardResponse;
import software.amazon.awssdk.services.cloudwatch.model.PutMetricDataResponse;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Scanner;
import java.util.concurrent.CompletableFuture;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including 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", "-");

    static CloudWatchActions cwActions = new CloudWatchActions();

    private static final Logger logger =
 LoggerFactory.getLogger(CloudWatchScenario.class);
    static Scanner scanner = new Scanner(System.in);
    public static void main(String[] args) throws Throwable {

        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/Billing 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 jsonWidgets.json in javav2/example_code/cloudwatch.)\s
             dashboardAdd - The location of a JSON file to use to update a
dashboard. (See CloudDashboard.json in javav2/example_code/cloudwatch.)\s
             settings - The location of a JSON file from which various values are
read. (See settings.json in javav2/example_code/cloudwatch.)\s
             metricImage - The location of a BMP file that is used to create a
graph.\s
           """;

       if (args.length != 7) {
           logger.info(usage);
           return;
       }
       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];

       logger.info(DASHES);
       logger.info("Welcome to the Amazon CloudWatch Basics scenario.");
       logger.info("""
           Amazon CloudWatch is a comprehensive monitoring and observability
service
           provided by Amazon Web Services (AWS). It is designed to help you
monitor your
           AWS resources, applications, and services, as well as on-premises
resources,
           in real-time.

           CloudWatch collects and tracks various types of data, including
metrics,
```

```
            logs, and events, from your AWS and on-premises resources. It allows you
to set
            alarms and automatically respond to changes in your environment,
            enabling you to quickly identify and address issues before they impact
your
            applications or services.

            With CloudWatch, you can gain visibility into your entire
infrastructure, from the cloud
            to the edge, and use this information to make informed decisions and
optimize your
            resource utilization.

            This scenario guides you through how to perform Amazon CloudWatch tasks
by using the
            AWS SDK for Java v2. Let's get started...
            """);
        waitForInputToContinue(scanner);

        try {
            runScenario(myDate, costDateWeek, dashboardName, dashboardJson,
dashboardAdd, settings, metricImage);
        } catch (RuntimeException e) {
            e.printStackTrace();
        }
        logger.info(DASHES);
    }

    private static void runScenario(String myDate, String costDateWeek, String
dashboardName, String dashboardJson, String dashboardAdd, String settings, String
metricImage ) throws Throwable {
        Double dataPoint = Double.parseDouble("10.0");
        logger.info(DASHES);
        logger.info("""
        1. List at least five available unique namespaces from Amazon CloudWatch.
        Select one from the list.
        """);
        String selectedNamespace;
        String selectedMetrics;
        int num;
        try {
            CompletableFuture<ArrayList<String>> future =
cwActions.listNameSpacesAsync();
            ArrayList<String> list = future.join();
```

```
            for (int z = 0; z < 5; z++) {
                int index = z + 1;
                logger.info("    " + index + ". {}", list.get(z));
            }

            num = Integer.parseInt(scanner.nextLine());
            if (1 <= num && num <= 5) {
                selectedNamespace = list.get(num - 1);
            } else {
                logger.info("You did not select a valid option.");
                return;
            }
            logger.info("You selected {}", selectedNamespace);

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof CloudWatchException cwEx) {
                logger.info("CloudWatch error occurred: Error message: {}, Error
code {}", cwEx.getMessage(), cwEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("2. List available metrics within the selected namespace.");
        logger.info("""
            A metric is a measure of the performance or health of your AWS
resources,
            applications, or custom resources. Metrics are the basic building blocks
of CloudWatch
            and provide data points that represent a specific aspect of your system
or application over time.

            Select a metric from the list.
            """);

        Dimension myDimension = null;
        try {
            CompletableFuture<ArrayList<String>> future =
cwActions.listMetsAsync(selectedNamespace);
```

```
            ArrayList<String> metList = future.join();
            logger.info("Metrics successfully retrieved. Total metrics: {}",
metList.size());
            for (int z = 0; z < 5; z++) {
                int index = z + 1;
                logger.info("    " + index + ". " + metList.get(z));
            }
            num = Integer.parseInt(scanner.nextLine());
            if (1 <= num && num <= 5) {
                selectedMetrics = metList.get(num - 1);
            } else {
                logger.info("You did not select a valid option.");
                return;
            }
            logger.info("You selected {}", selectedMetrics);

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof CloudWatchException cwEx) {
                logger.info("CloudWatch error occurred: Error message: {}, Error
code {}", cwEx.getMessage(), cwEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }

        try {
            myDimension = cwActions.getSpecificMetAsync(selectedNamespace).join();
            logger.info("Metric statistics successfully retrieved and displayed.");
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof CloudWatchException cwEx) {
                logger.info("CloudWatch error occurred: Error message: {}, Error
code {}", cwEx.getMessage(), cwEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }

        waitForInputToContinue(scanner);
        logger.info(DASHES);
```

```
        logger.info(DASHES);
        logger.info("3. Get statistics for the selected metric over the last day.");
        logger.info("""
            Statistics refer to the various mathematical calculations that can be
performed on the
            collected metrics to derive meaningful insights. Statistics provide a
way to summarize and
            analyze the data collected for a specific metric over a specified time
period.
            """);
        waitForInputToContinue(scanner);
        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++) {
            logger.info("    " + (t + 1) + ". {}", statTypes.get(t));
        }
        logger.info("Select a metric statistic by entering a number from the
preceding list:");
        num = Integer.parseInt(scanner.nextLine());
        if (1 <= num && num <= 5) {
            metricOption = statTypes.get(num - 1);
        } else {
            logger.info("You did not select a valid option.");
            return;
        }
        logger.info("You selected " + metricOption);
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<GetMetricStatisticsResponse> future =
cwActions.getAndDisplayMetricStatisticsAsync(selectedNamespace, selectedMetrics,
metricOption, myDate, myDimension);
            future.join();
            logger.info("Metric statistics retrieved successfully.");

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof CloudWatchException cwEx) {
```

```
                logger.info("CloudWatch error occurred: Error message: {}, Error
code {}", cwEx.getMessage(), cwEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("4. Get CloudWatch estimated billing for the last week.");
        waitForInputToContinue(scanner);
         try {
            CompletableFuture<GetMetricStatisticsResponse> future =
cwActions.getMetricStatisticsAsync(costDateWeek);
            future.join();

            logger.info("Metric statistics successfully retrieved and displayed.");
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof CloudWatchException cwEx) {
                logger.info("CloudWatch error occurred: Error message: {}, Error
code {}", cwEx.getMessage(), cwEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
             throw cause;
         }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("5. Create a new CloudWatch dashboard with metrics.");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<PutDashboardResponse> future =
cwActions.createDashboardWithMetricsAsync(dashboardName, dashboardJson);
            future.join();

        } catch (RuntimeException | IOException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof DashboardInvalidInputErrorException cwEx) {
```

```
                logger.info("Invalid CloudWatch data. Error message: {}, Error code
{}", cwEx.getMessage(), cwEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("6. List dashboards using a paginator.");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Void> future = cwActions.listDashboardsAsync();
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof CloudWatchException cwEx) {
                logger.info("CloudWatch error occurred: Error message: {}, Error
code {}", cwEx.getMessage(), cwEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("7. Create a new custom metric by adding data to it.");
        logger.info("""
            The primary benefit of using a custom metric in Amazon CloudWatch is the
ability to
            monitor and collect data that is specific to your application or
infrastructure.
            """);
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<PutMetricDataResponse> future =
cwActions.createNewCustomMetricAsync(dataPoint);
            future.join();
```

```
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof CloudWatchException cwEx) {
                logger.info("CloudWatch error occurred: Error message: {}, Error
code {}", cwEx.getMessage(), cwEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("8. Add an additional metric to the dashboard.");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<PutDashboardResponse> future =
cwActions.addMetricToDashboardAsync(dashboardAdd, dashboardName);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof DashboardInvalidInputErrorException cwEx) {
                logger.info("Invalid CloudWatch data. Error message: {}, Error code
{}", cwEx.getMessage(), cwEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("9. Create an alarm for the custom metric.");
        waitForInputToContinue(scanner);
        String alarmName = "" ;
        try {
            CompletableFuture<String> future = cwActions.createAlarmAsync(settings);
            alarmName = future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof LimitExceededException cwEx) {
```

```
                logger.info("The quota for alarms has been reached: Error message:
{}, Error code {}", cwEx.getMessage(), cwEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("10. Describe ten current alarms.");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Void> future = cwActions.describeAlarmsAsync();
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof CloudWatchException cwEx) {
                logger.info("CloudWatch error occurred: Error message: {}, Error
code {}", cwEx.getMessage(), cwEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("11. Get current data for new custom metric.");
        try {
            CompletableFuture<Void> future =
cwActions.getCustomMetricDataAsync(settings);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof CloudWatchException cwEx) {
                logger.info("CloudWatch error occurred: Error message: {}, Error
code {}", cwEx.getMessage(), cwEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
```

```
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("12. Push data into the custom metric to trigger the alarm.");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<PutMetricDataResponse> future =
cwActions.addMetricDataForAlarmAsync(settings);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof CloudWatchException cwEx) {
                logger.info("CloudWatch error occurred: Error message: {}, Error
code {}", cwEx.getMessage(), cwEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("13. Check the alarm state using the action
DescribeAlarmsForMetric.");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Void> future =
cwActions.checkForMetricAlarmAsync(settings);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof CloudWatchException cwEx) {
                logger.info("CloudWatch error occurred: Error message: {}, Error
code {}", cwEx.getMessage(), cwEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
```

```
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("14. Get alarm history for the new alarm.");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Void> future =
cwActions.getAlarmHistoryAsync(settings, myDate);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof CloudWatchException cwEx) {
                logger.info("CloudWatch error occurred: Error message: {}, Error
code {}", cwEx.getMessage(), cwEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("15. Add an anomaly detector for the custom metric.");
        logger.info("""
            An anomaly detector is a feature that automatically detects unusual
patterns or deviations in your
            monitored metrics. It uses machine learning algorithms to analyze the
historical behavior
            of your metrics and establish a baseline.

            The anomaly detector then compares the current metric values against
this baseline and
            identifies any anomalies or outliers that may indicate potential issues
or unexpected changes
            in your system's performance or behavior.

            """);
        waitForInputToContinue(scanner);
        try {
```

```
            CompletableFuture<Void> future =
cwActions.addAnomalyDetectorAsync(settings);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof CloudWatchException cwEx) {
                logger.info("CloudWatch error occurred: Error message: {}, Error
code {}", cwEx.getMessage(), cwEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("16. Describe current anomaly detectors.");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Void> future =
cwActions.describeAnomalyDetectorsAsync(settings);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof CloudWatchException cwEx) {
                logger.info("CloudWatch error occurred: Error message: {}, Error
code {}", cwEx.getMessage(), cwEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("17. Get a metric image for the custom metric.");
        try {
            CompletableFuture<Void> future =
cwActions.downloadAndSaveMetricImageAsync(metricImage);
            future.join();
```

```
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof CloudWatchException cwEx) {
                logger.info("CloudWatch error occurred: Error message: {}, Error
code {}", cwEx.getMessage(), cwEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("18. Clean up the Amazon CloudWatch resources.");

        try {
            logger.info(". Delete the Dashboard.");
            waitForInputToContinue(scanner);
            CompletableFuture<DeleteDashboardsResponse> future =
cwActions.deleteDashboardAsync(dashboardName);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof CloudWatchException cwEx) {
                logger.info("CloudWatch error occurred: Error message: {}, Error
code {}", cwEx.getMessage(), cwEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }

        try {
            logger.info("Delete the alarm.");
            waitForInputToContinue(scanner);
            CompletableFuture<DeleteAlarmsResponse> future =
cwActions.deleteCWAlarmAsync(alarmName);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof CloudWatchException cwEx) {
```

```
                logger.info("CloudWatch error occurred: Error message: {}, Error
code {}", cwEx.getMessage(), cwEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }

        try {
            logger.info("Delete the anomaly detector.");
            waitForInputToContinue(scanner);
            CompletableFuture<DeleteAnomalyDetectorResponse> future =
cwActions.deleteAnomalyDetectorAsync(settings);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof CloudWatchException cwEx) {
                logger.info("CloudWatch error occurred: Error message: {}, Error
code {}", cwEx.getMessage(), cwEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("The Amazon CloudWatch example scenario is complete.");
        logger.info(DASHES);
    }

    private static void waitForInputToContinue(Scanner scanner) {
        while (true) {
            logger.info("");
            logger.info("Enter 'c' followed by <ENTER> to continue:");
            String input = scanner.nextLine();
            if (input.trim().equalsIgnoreCase("c")) {
                logger.info("Continuing with the program...");
                logger.info("");
                break;
            } else {
                // Handle invalid input.
```

```
                logger.info("Invalid input. Please try again.");
            }
        }
    }
}
```

Una classe wrapper per i metodi SDK. CloudWatch

```
public class CloudWatchActions {

    private static CloudWatchAsyncClient cloudWatchAsyncClient;

    private static final Logger logger =
 LoggerFactory.getLogger(CloudWatchActions.class);

    /**
     * Retrieves an asynchronous CloudWatch client instance.
     *
     * <p>
     * This method ensures that the CloudWatch client is initialized with the
 following configurations:
     * <ul>
     *     <li>Maximum concurrency: 100</li>
     *     <li>Connection timeout: 60 seconds</li>
     *     <li>Read timeout: 60 seconds</li>
     *     <li>Write timeout: 60 seconds</li>
     *     <li>API call timeout: 2 minutes</li>
     *     <li>API call attempt timeout: 90 seconds</li>
     *     <li>Retry strategy: STANDARD</li>
     * </ul>
     * </p>
     *
     * @return the asynchronous CloudWatch client instance
     */
    private static CloudWatchAsyncClient getAsyncClient() {
        if (cloudWatchAsyncClient == null) {
            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                .maxConcurrency(100)
                .connectionTimeout(Duration.ofSeconds(60))
                .readTimeout(Duration.ofSeconds(60))
                .writeTimeout(Duration.ofSeconds(60))
                .build();
```

```
            ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
                .apiCallTimeout(Duration.ofMinutes(2))
                .apiCallAttemptTimeout(Duration.ofSeconds(90))
                .retryStrategy(RetryMode.STANDARD)
                .build();

            cloudWatchAsyncClient = CloudWatchAsyncClient.builder()
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return cloudWatchAsyncClient;
    }

    /**
     * Deletes an Anomaly Detector.
     *
     * @param fileName the name of the file containing the Anomaly Detector
configuration
     * @return a CompletableFuture that represents the asynchronous deletion of the
Anomaly Detector
     */
    public CompletableFuture<DeleteAnomalyDetectorResponse>
deleteAnomalyDetectorAsync(String fileName) {
        CompletableFuture<JsonNode> readFileFuture =
CompletableFuture.supplyAsync(() -> {
            try {
                JsonParser parser = new JsonFactory().createParser(new
File(fileName));
                return new ObjectMapper().readTree(parser); // Return the root node
            } catch (IOException e) {
                throw new RuntimeException("Failed to read or parse the file", e);
            }
        });

        return readFileFuture.thenCompose(rootNode -> {
            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();

            return getAsyncClient().deleteAnomalyDetector(request);
        }).whenComplete((result, exception) -> {
            if (exception != null) {
                throw new RuntimeException("Failed to delete the Anomaly Detector",
exception);
            } else {
                logger.info("Successfully deleted the Anomaly Detector.");
            }
        });
    }

    /**
     * Deletes a CloudWatch alarm.
     *
     * @param alarmName the name of the alarm to be deleted
     * @return a {@link CompletableFuture} representing the asynchronous operation
to delete the alarm
     * the {@link DeleteAlarmsResponse} is returned when the operation completes
successfully,
     * or a {@link RuntimeException} is thrown if the operation fails
     */
    public CompletableFuture<DeleteAlarmsResponse> deleteCWAlarmAsync(String
alarmName) {
        DeleteAlarmsRequest request = DeleteAlarmsRequest.builder()
            .alarmNames(alarmName)
            .build();

        return getAsyncClient().deleteAlarms(request)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    throw new RuntimeException("Failed to delete the alarm:{} " +
alarmName, exception);
```

```
                } else {
                    logger.info("Successfully deleted alarm {} ", alarmName);
                }
            });
    }

    /**
     * Deletes the specified dashboard.
     *
     * @param dashboardName the name of the dashboard to be deleted
     * @return a {@link CompletableFuture} representing the asynchronous operation
of deleting the dashboard
     * @throws RuntimeException if the dashboard deletion fails
     */
    public CompletableFuture<DeleteDashboardsResponse> deleteDashboardAsync(String
dashboardName) {
        DeleteDashboardsRequest dashboardsRequest =
DeleteDashboardsRequest.builder()
            .dashboardNames(dashboardName)
            .build();

        return getAsyncClient().deleteDashboards(dashboardsRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    throw new RuntimeException("Failed to delete the dashboard: " +
dashboardName, exception);
                } else {
                    logger.info("{} was successfully deleted.", dashboardName);
                }
            });
    }


    /**
     * Retrieves and saves a custom metric image to a file.
     *
     * @param fileName the name of the file to save the metric image to
     * @return a {@link CompletableFuture} that completes when the image has been
saved to the file
     */
    public CompletableFuture<Void> downloadAndSaveMetricImageAsync(String fileName)
{
        logger.info("Getting Image data for custom metric.");
        String myJSON = """
```

```
              {
                  "title": "Example Metric Graph",
                  "view": "timeSeries",
                  "stacked ": false,
                  "period": 10,
                  "width": 1400,
                  "height": 600,
                  "metrics": [
                      [
                      "AWS/Billing",
                      "EstimatedCharges",
                      "Currency",
                      "USD"
                     ]
                  ]
              }
             """;

       GetMetricWidgetImageRequest imageRequest =
GetMetricWidgetImageRequest.builder()
            .metricWidget(myJSON)
            .build();

       return getAsyncClient().getMetricWidgetImage(imageRequest)
            .thenCompose(response -> {
                SdkBytes sdkBytes = response.metricWidgetImage();
                byte[] bytes = sdkBytes.asByteArray();
                return CompletableFuture.runAsync(() -> {
                    try {
                        File outputFile = new File(fileName);
                        try (FileOutputStream outputStream = new
FileOutputStream(outputFile)) {
                            outputStream.write(bytes);
                        }
                    } catch (IOException e) {
                        throw new RuntimeException("Failed to write image to file",
e);
                    }
                });
            })
            .whenComplete((result, exception) -> {
                if (exception != null) {
                    throw new RuntimeException("Error getting and saving metric
image", exception);
```

```
                } else {
                    logger.info("Image data saved successfully to {}", fileName);
                }
            });
    }


    /**
     * Describes the anomaly detectors based on the specified JSON file.
     *
     * @param fileName the name of the JSON file containing the custom metric
namespace and name
     * @return a {@link CompletableFuture} that completes when the anomaly detectors
have been described
     * @throws RuntimeException if there is a failure during the operation, such as
when reading or parsing the JSON file,
     *                          or when describing the anomaly detectors
     */
    public CompletableFuture<Void> describeAnomalyDetectorsAsync(String fileName) {
        CompletableFuture<JsonNode> readFileFuture =
CompletableFuture.supplyAsync(() -> {
            try {
                JsonParser parser = new JsonFactory().createParser(new
File(fileName));
                return new ObjectMapper().readTree(parser);
            } catch (IOException e) {
                throw new RuntimeException("Failed to read or parse the file", e);
            }
        });

        return readFileFuture.thenCompose(rootNode -> {
            try {
                String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
                String customMetricName =
rootNode.findValue("customMetricName").asText();

                DescribeAnomalyDetectorsRequest detectorsRequest =
DescribeAnomalyDetectorsRequest.builder()
                    .maxResults(10)
                    .metricName(customMetricName)
                    .namespace(customMetricNamespace)
                    .build();
```

```
                return
getAsyncClient().describeAnomalyDetectors(detectorsRequest).thenAccept(response ->
{
                    List<AnomalyDetector> anomalyDetectorList =
response.anomalyDetectors();
                    for (AnomalyDetector detector : anomalyDetectorList) {
                        logger.info("Metric name: {} ",
detector.singleMetricAnomalyDetector().metricName());
                        logger.info("State: {} ", detector.stateValue());
                    }
                });
            } catch (RuntimeException e) {
                throw new RuntimeException("Failed to describe anomaly detectors",
e);
            }
        }).whenComplete((result, exception) -> {
            if (exception != null) {
                throw new RuntimeException("Error describing anomaly detectors",
exception);
            }
        });
    }


    /**
     * Adds an anomaly detector for the given file.
     *
     * @param fileName the name of the file containing the anomaly detector
configuration
     * @return a {@link CompletableFuture} that completes when the anomaly detector
has been added
     */
    public CompletableFuture<Void> addAnomalyDetectorAsync(String fileName) {
        CompletableFuture<JsonNode> readFileFuture =
CompletableFuture.supplyAsync(() -> {
            try {
                JsonParser parser = new JsonFactory().createParser(new
File(fileName));
                return new ObjectMapper().readTree(parser); // Return the root node
            } catch (IOException e) {
                throw new RuntimeException("Failed to read or parse the file", e);
            }
        });
```

```
        return readFileFuture.thenCompose(rootNode -> {
            try {
                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();

                return
getAsyncClient().putAnomalyDetector(anomalyDetectorRequest).thenAccept(response ->
{
                    logger.info("Added anomaly detector for metric {}",
customMetricName);
                });
            } catch (Exception e) {
                throw new RuntimeException("Failed to create anomaly detector", e);
            }
        }).whenComplete((result, exception) -> {
            if (exception != null) {
                throw new RuntimeException("Error adding anomaly detector",
exception);
            }
        });
    }


    /**
     * Retrieves the alarm history for a given alarm name and date range.
     *
     * @param fileName the path to the JSON file containing the alarm name
     * @param date     the date to start the alarm history search (in the format
"yyyy-MM-dd'T'HH:mm:ss'Z'")
```

```
    * @return a {@code CompletableFuture<Void>} that completes when the alarm
history has been retrieved and processed
    */
   public CompletableFuture<Void> getAlarmHistoryAsync(String fileName, String
date) {
       CompletableFuture<String> readFileFuture = CompletableFuture.supplyAsync(()
-> {
           try {
               JsonParser parser = new JsonFactory().createParser(new
File(fileName));
               com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
               return rootNode.findValue("exampleAlarmName").asText(); // Return
alarmName from the JSON file
           } catch (IOException e) {
               throw new RuntimeException("Failed to read or parse the file", e);
           }
       });

       // Use the alarm name to describe alarm history with a paginator.
       return readFileFuture.thenCompose(alarmName -> {
           try {
               Instant start = Instant.parse(date);
               Instant endDate = Instant.now();
               DescribeAlarmHistoryRequest historyRequest =
DescribeAlarmHistoryRequest.builder()
                   .startDate(start)
                   .endDate(endDate)
                   .alarmName(alarmName)
                   .historyItemType(HistoryItemType.ACTION)
                   .build();

               // Use the paginator to paginate through alarm history pages.
               DescribeAlarmHistoryPublisher historyPublisher =
getAsyncClient().describeAlarmHistoryPaginator(historyRequest);
               CompletableFuture<Void> future = historyPublisher
                   .subscribe(response -> response.alarmHistoryItems().forEach(item
-> {
                       logger.info("History summary: {}", item.historySummary());
                       logger.info("Timestamp: {}", item.timestamp());
                   }))
                   .whenComplete((result, exception) -> {
                       if (exception != null) {
```

```
                            logger.error("Error occurred while getting alarm
history: " + exception.getMessage(), exception);
                        } else {
                            logger.info("Successfully retrieved all alarm
history.");
                        }
                    });

                // Return the future to the calling code for further handling
                return future;
            } catch (Exception e) {
                throw new RuntimeException("Failed to process alarm history", e);
            }
        }).whenComplete((result, exception) -> {
            if (exception != null) {
                throw new RuntimeException("Error completing alarm history
processing", exception);
            }
        });
    }



    /**
     * Checks for a metric alarm in AWS CloudWatch.
     *
     * @param fileName the name of the file containing the JSON configuration for
the custom metric
     * @return a {@link CompletableFuture} that completes when the check for the
metric alarm is complete
     */
    public CompletableFuture<Void> checkForMetricAlarmAsync(String fileName) {
        CompletableFuture<String> readFileFuture = CompletableFuture.supplyAsync(()
-> {
            try {
                JsonParser parser = new JsonFactory().createParser(new
File(fileName));
                com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
                return rootNode.toString(); // Return JSON as a string for further
processing
            } catch (IOException e) {
                throw new RuntimeException("Failed to read file", e);
            }
```

```
        });

        return readFileFuture.thenCompose(jsonContent -> {
            try {
                com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(jsonContent);
                String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
                String customMetricName =
rootNode.findValue("customMetricName").asText();

                DescribeAlarmsForMetricRequest metricRequest =
DescribeAlarmsForMetricRequest.builder()
                    .metricName(customMetricName)
                    .namespace(customMetricNamespace)
                    .build();

                return checkForAlarmAsync(metricRequest, customMetricName, 10);

            } catch (IOException e) {
                throw new RuntimeException("Failed to parse JSON content", e);
            }
        }).whenComplete((result, exception) -> {
            if (exception != null) {
                throw new RuntimeException("Error checking metric alarm",
exception);
            }
        });
    }

    // Recursive method to check for the alarm.

    /**
     * Checks for the existence of an alarm asynchronously for the specified metric.
     *
     * @param metricRequest    the request to describe the alarms for the specified
metric
     * @param customMetricName the name of the custom metric to check for an alarm
     * @param retries          the number of retries to perform if no alarm is found
     * @return a {@link CompletableFuture} that completes when an alarm is found or
the maximum number of retries has been reached
     */
```

```
   private static CompletableFuture<Void>
checkForAlarmAsync(DescribeAlarmsForMetricRequest metricRequest, String
customMetricName, int retries) {
       if (retries == 0) {
           return CompletableFuture.completedFuture(null).thenRun(() ->
               logger.info("No Alarm state found for {} after 10 retries.",
customMetricName)
           );
       }

       return
(getAsyncClient().describeAlarmsForMetric(metricRequest).thenCompose(response -> {
           if (response.hasMetricAlarms()) {
               logger.info("Alarm state found for {}", customMetricName);
               return CompletableFuture.completedFuture(null); // Alarm found,
complete the future
           } else {
               return CompletableFuture.runAsync(() -> {
                   try {
                       Thread.sleep(20000);
                       logger.info(".");
                   } catch (InterruptedException e) {
                       throw new RuntimeException("Interrupted while waiting to
retry", e);
                   }
               }).thenCompose(v -> checkForAlarmAsync(metricRequest,
customMetricName, retries - 1)); // Recursive call
           }
       }));
   }


   /**
    * Adds metric data for an alarm asynchronously.
    *
    * @param fileName the name of the JSON file containing the metric data
    * @return a CompletableFuture that asynchronously returns the
PutMetricDataResponse
    */
   public CompletableFuture<PutMetricDataResponse>
addMetricDataForAlarmAsync(String fileName) {
       CompletableFuture<String> readFileFuture = CompletableFuture.supplyAsync(()
-> {
           try {
```

```
                JsonParser parser = new JsonFactory().createParser(new
File(fileName));
                com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
                return rootNode.toString(); // Return JSON as a string for further
processing
            } catch (IOException e) {
                throw new RuntimeException("Failed to read file", e);
            }
        });

        return readFileFuture.thenCompose(jsonContent -> {
            try {
                com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(jsonContent);
                String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
                String customMetricName =
rootNode.findValue("customMetricName").asText();
                Instant instant = Instant.now();

                // Create MetricDatum objects.
                MetricDatum datum1 = 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(datum1);
                metricDataList.add(datum2);

                // Build the PutMetricData request.
                PutMetricDataRequest request = PutMetricDataRequest.builder()
                    .namespace(customMetricNamespace)
                    .metricData(metricDataList)
```

```
                    .build();

                // Send the request asynchronously.
                return getAsyncClient().putMetricData(request);

            } catch (IOException e) {
                CompletableFuture<PutMetricDataResponse> failedFuture = new
CompletableFuture<>();
                failedFuture.completeExceptionally(new RuntimeException("Failed to
parse JSON content", e));
                return failedFuture;
            }
        }).whenComplete((response, exception) -> {
            if (exception != null) {
                logger.error("Failed to put metric data: " + exception.getMessage(),
exception);
            } else {
                logger.info("Added metric values for metric.");
            }
        });
    }


    /**
     * Retrieves custom metric data from the AWS CloudWatch service.
     *
     * @param fileName the name of the file containing the custom metric information
     * @return a {@link CompletableFuture} that completes when the metric data has
been retrieved
     */
    public CompletableFuture<Void> getCustomMetricDataAsync(String fileName) {
        CompletableFuture<String> readFileFuture = CompletableFuture.supplyAsync(()
-> {
            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);
                return rootNode.toString(); // Return JSON as a string for further
processing
            } catch (IOException e) {
                throw new RuntimeException("Failed to read file", e);
            }
```

```
        });

        return readFileFuture.thenCompose(jsonContent -> {
            try {
                // Parse the JSON string to extract relevant values.
                com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(jsonContent);
                String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
                String customMetricName =
rootNode.findValue("customMetricName").asText();

                // Set the current time and date range for metric query.
                Instant nowDate = Instant.now();
                long hours = 1;
                long minutes = 30;
                Instant endTime = 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(60)  // Assuming period in seconds
                    .metric(met)
                    .build();

                MetricDataQuery dataQuery = MetricDataQuery.builder()
                    .metricStat(metStat)
                    .id("foo2")
                    .returnData(true)
                    .build();

                List<MetricDataQuery> dq = new ArrayList<>();
                dq.add(dataQuery);

                GetMetricDataRequest getMetricDataRequest =
GetMetricDataRequest.builder()
                    .maxDatapoints(10)
                    .scanBy(ScanBy.TIMESTAMP_DESCENDING)
                    .startTime(nowDate)
```

```
                    .endTime(endTime)
                    .metricDataQueries(dq)
                    .build();

                // Call the async method for CloudWatch data retrieval.
                return getAsyncClient().getMetricData(getMetricDataRequest);

            } catch (IOException e) {
                throw new RuntimeException("Failed to parse JSON content", e);
            }
        }).thenAccept(response -> {
            List<MetricDataResult> data = response.metricDataResults();
            for (MetricDataResult item : data) {
                logger.info("The label is: {}", item.label());
                logger.info("The status code is: {}", item.statusCode().toString());
            }
        }).exceptionally(exception -> {
            throw new RuntimeException("Failed to get metric data", exception);
        });
    }


    /**
     * Describes the CloudWatch alarms of the 'METRIC_ALARM' type.
     *
     * @return a {@link CompletableFuture} that represents the asynchronous
operation
     * of describing the CloudWatch alarms. The future completes when the
     * operation is finished, either successfully or with an error.
     */
    public CompletableFuture<Void> describeAlarmsAsync() {
        List<AlarmType> typeList = new ArrayList<>();
        typeList.add(AlarmType.METRIC_ALARM);
        DescribeAlarmsRequest alarmsRequest = DescribeAlarmsRequest.builder()
            .alarmTypes(typeList)
            .maxRecords(10)
            .build();

        return getAsyncClient().describeAlarms(alarmsRequest)
            .thenAccept(response -> {
                List<MetricAlarm> alarmList = response.metricAlarms();
                for (MetricAlarm alarm : alarmList) {
                    logger.info("Alarm name: {}", alarm.alarmName());
                    logger.info("Alarm description: {} ", alarm.alarmDescription());
```

```
                }
            })
            .whenComplete((response, ex) -> {
                if (ex != null) {
                    logger.info("Failed to describe alarms: {}", ex.getMessage());
                } else {
                    logger.info("Successfully described alarms.");
                }
            });
    }

    /**
     * Creates an alarm based on the configuration provided in a JSON file.
     *
     * @param fileName the name of the JSON file containing the alarm configuration
     * @return a CompletableFuture that represents the asynchronous operation of
creating the alarm
     * @throws RuntimeException if an exception occurs while reading the JSON file
or creating the alarm
     */
    public CompletableFuture<String> createAlarmAsync(String fileName) {
        com.fasterxml.jackson.databind.JsonNode rootNode;
        try {
            JsonParser parser = new JsonFactory().createParser(new File(fileName));
            rootNode = new ObjectMapper().readTree(parser);
        } catch (IOException e) {
            throw new RuntimeException("Failed to read the alarm configuration
file", e);
        }

        // Extract values from the JSON node.
        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();

        // Call the putMetricAlarm asynchronously and handle the result.
        return getAsyncClient().putMetricAlarm(alarmRequest)
            .handle((response, ex) -> {
                if (ex != null) {
                    logger.info("Failed to create alarm: {}", ex.getMessage());
                    throw new RuntimeException("Failed to create alarm", ex);
                } else {
                    logger.info("{} was successfully created!", alarmName);
                    return alarmName;
                }
            });
    }

    /**
     * Adds a metric to a dashboard asynchronously.
     *
     * @param fileName      the name of the file containing the dashboard content
     * @param dashboardName the name of the dashboard to be updated
     * @return a {@link CompletableFuture} representing the asynchronous operation,
which will complete with a
     * {@link PutDashboardResponse} when the dashboard is successfully updated
     */
    public CompletableFuture<PutDashboardResponse> addMetricToDashboardAsync(String
fileName, String dashboardName) {
        String dashboardBody;
        try {
            dashboardBody = readFileAsString(fileName);
        } catch (IOException e) {
```

```
            throw new RuntimeException("Failed to read the dashboard file", e);
        }

        PutDashboardRequest dashboardRequest = PutDashboardRequest.builder()
            .dashboardName(dashboardName)
            .dashboardBody(dashboardBody)
            .build();

        return getAsyncClient().putDashboard(dashboardRequest)
            .handle((response, ex) -> {
                if (ex != null) {
                    logger.info("Failed to update dashboard: {}", ex.getMessage());
                    throw new RuntimeException("Error updating dashboard", ex);
                } else {
                    logger.info("{} was successfully updated.", dashboardName);
                    return response;
                }
            });
    }

    /**
     * Creates a new custom metric.
     *
     * @param dataPoint the data point to be added to the custom metric
     * @return a {@link CompletableFuture} representing the asynchronous operation
of adding the custom metric
     */
    public CompletableFuture<PutMetricDataResponse>
createNewCustomMetricAsync(Double dataPoint) {
        Dimension dimension = Dimension.builder()
            .name("UNIQUE_PAGES")
            .value("URLS")
            .build();

        // Set an Instant object for the current time in UTC.
        String time =
ZonedDateTime.now(ZoneOffset.UTC).format(DateTimeFormatter.ISO_INSTANT);
        Instant instant = Instant.parse(time);

        // Create the MetricDatum.
        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();

        return getAsyncClient().putMetricData(request)
            .whenComplete((response, ex) -> {
                if (ex != null) {
                    throw new RuntimeException("Error adding custom metric", ex);
                } else {
                    logger.info("Successfully added metric values for
PAGES_VISITED.");
                }
            });
    }

    /**
     * Lists the available dashboards.
     *
     * @return a {@link CompletableFuture} that completes when the operation is
finished.
     * The future will complete exceptionally if an error occurs while listing the
dashboards.
     */
    public CompletableFuture<Void> listDashboardsAsync() {
        ListDashboardsRequest listDashboardsRequest =
ListDashboardsRequest.builder().build();
        ListDashboardsPublisher paginator =
getAsyncClient().listDashboardsPaginator(listDashboardsRequest);
        return paginator.subscribe(response -> {
            response.dashboardEntries().forEach(entry -> {
                logger.info("Dashboard name is: {} ", entry.dashboardName());
                logger.info("Dashboard ARN is: {} ", entry.dashboardArn());
            });
        }).exceptionally(ex -> {
            logger.info("Failed to list dashboards: {} ", ex.getMessage());
            throw new RuntimeException("Error occurred while listing dashboards",
ex);
        });
    }
```

```
   /**
    * Creates a new dashboard with the specified name and metrics from the given
file.
    *
    * @param dashboardName the name of the dashboard to be created
    * @param fileName      the name of the file containing the dashboard body
    * @return a {@link CompletableFuture} representing the asynchronous operation
of creating the dashboard
    * @throws IOException if there is an error reading the dashboard body from the
file
    */
   public CompletableFuture<PutDashboardResponse>
createDashboardWithMetricsAsync(String dashboardName, String fileName) throws
IOException {
       String dashboardBody = readFileAsString(fileName);
       PutDashboardRequest dashboardRequest = PutDashboardRequest.builder()
           .dashboardName(dashboardName)
           .dashboardBody(dashboardBody)
           .build();

       return getAsyncClient().putDashboard(dashboardRequest)
           .handle((response, ex) -> {
               if (ex != null) {
                   logger.info("Failed to create dashboard: {}", ex.getMessage());
                   throw new RuntimeException("Dashboard creation failed", ex);
               } else {
                   // Handle the normal response case
                   logger.info("{} was successfully created.", dashboardName);
                   List<DashboardValidationMessage> messages =
response.dashboardValidationMessages();
                   if (messages.isEmpty()) {
                       logger.info("There are no messages in the new Dashboard.");
                   } else {
                       for (DashboardValidationMessage message : messages) {
                           logger.info("Message: {}", message.message());
                       }
                   }
                   return response; // Return the response for further use
               }
           });
   }
```

```
   /**
    * Retrieves the metric statistics for the "EstimatedCharges" metric in the
"AWS/Billing" namespace.
    *
    * @param costDateWeek the start date for the metric statistics, in the format
of an ISO-8601 date string (e.g., "2023-04-05")
    * @return a {@link CompletableFuture} that, when completed, contains the {@link
GetMetricStatisticsResponse} with the retrieved metric statistics
    * @throws RuntimeException if the metric statistics cannot be retrieved
successfully
    */
   public CompletableFuture<GetMetricStatisticsResponse>
getMetricStatisticsAsync(String costDateWeek) {
       Instant start = Instant.parse(costDateWeek);
       Instant endDate = Instant.now();

       // Define dimension
       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) // One day period
           .build();

       return getAsyncClient().getMetricStatistics(statisticsRequest)
           .whenComplete((response, exception) -> {
               if (response != null) {
                   List<Datapoint> data = response.datapoints();
                   if (!data.isEmpty()) {
                       for (Datapoint datapoint : data) {
```

```
                            logger.info("Timestamp: {} Maximum value: {})",
datapoint.timestamp(), datapoint.maximum());
                        }
                    } else {
                        logger.info("The returned data list is empty");
                    }
                } else {
                    throw new RuntimeException("Failed to get metric statistics: " +
exception.getMessage(), exception);
                }
            });
    }


    /**
     * Retrieves and displays metric statistics for the specified parameters.
     *
     * @param nameSpace    the namespace for the metric
     * @param metVal       the name of the metric
     * @param metricOption the statistic to retrieve for the metric (e.g.,
"Maximum", "Average")
     * @param date         the date for which to retrieve the metric statistics, in
the format "yyyy-MM-dd'T'HH:mm:ss'Z'"
     * @param myDimension  the dimension(s) to filter the metric statistics by
     * @return a {@link CompletableFuture} that completes when the metric statistics
have been retrieved and displayed
     */
    public CompletableFuture<GetMetricStatisticsResponse>
getAndDisplayMetricStatisticsAsync(String nameSpace, String metVal,

            String metricOption, String date, Dimension myDimension) {

        Instant start = Instant.parse(date);
        Instant endDate = Instant.now();

        // Building the request for metric statistics.
        GetMetricStatisticsRequest statisticsRequest =
GetMetricStatisticsRequest.builder()
                .endTime(endDate)
                .startTime(start)
                .dimensions(myDimension)
                .metricName(metVal)
                .namespace(nameSpace)
                .period(86400) // 1 day period
```

```
            .statistics(Statistic.fromValue(metricOption))
            .build();

        return getAsyncClient().getMetricStatistics(statisticsRequest)
            .whenComplete((response, exception) -> {
                if (response != null) {
                    List<Datapoint> data = response.datapoints();
                    if (!data.isEmpty()) {
                        for (Datapoint datapoint : data) {
                            logger.info("Timestamp: {} Maximum value: {}",
datapoint.timestamp(), datapoint.maximum());
                        }
                    } else {
                        logger.info("The returned data list is empty");
                    }
                } else {
                    logger.info("Failed to get metric statistics: {} ",
exception.getMessage());
                }
            })
            .exceptionally(exception -> {
                throw new RuntimeException("Error while getting metric statistics: "
+ exception.getMessage(), exception);
            });
    }


    /**
     * Retrieves a list of metric names for the specified namespace.
     *
     * @param namespace the namespace for which to retrieve the metric names
     * @return a {@link CompletableFuture} that, when completed, contains an {@link
ArrayList} of
     * the metric names in the specified namespace
     * @throws RuntimeException if an error occurs while listing the metrics
     */
    public CompletableFuture<ArrayList<String>> listMetsAsync(String namespace) {
        ListMetricsRequest request = ListMetricsRequest.builder()
            .namespace(namespace)
            .build();

        ListMetricsPublisher metricsPaginator =
getAsyncClient().listMetricsPaginator(request);
        Set<String> metSet = new HashSet<>();
```

```
        CompletableFuture<Void> future = metricsPaginator.subscribe(response -> {
            response.metrics().forEach(metric -> {
                String metricName = metric.metricName();
                metSet.add(metricName);
            });
        });

        return future
            .thenApply(ignored -> new ArrayList<>(metSet))
            .exceptionally(exception -> {
                throw new RuntimeException("Failed to list metrics: " +
exception.getMessage(), exception);
            });
    }

    /**
     * Lists the available namespaces for the current AWS account.
     *
     * @return a {@link CompletableFuture} that, when completed, contains an {@link
ArrayList} of the available namespace names.
     * @throws RuntimeException if an error occurs while listing the namespaces.
     */
    public CompletableFuture<ArrayList<String>> listNameSpacesAsync() {
        ArrayList<String> nameSpaceList = new ArrayList<>();
        ListMetricsRequest request = ListMetricsRequest.builder().build();

        ListMetricsPublisher metricsPaginator =
getAsyncClient().listMetricsPaginator(request);
        CompletableFuture<Void> future = metricsPaginator.subscribe(response -> {
            response.metrics().forEach(metric -> {
                String namespace = metric.namespace();
                if (!nameSpaceList.contains(namespace)) {
                    nameSpaceList.add(namespace);
                }
            });
        });

        return future
            .thenApply(ignored -> nameSpaceList)
            .exceptionally(exception -> {
                throw new RuntimeException("Failed to list namespaces: " +
exception.getMessage(), exception);
            });
    }
```

```
    /**
     * Retrieves the specific metric asynchronously.
     *
     * @param namespace the namespace of the metric to retrieve
     * @return a CompletableFuture that completes with the first dimension of the
 first metric found in the specified namespace,
     * or throws a RuntimeException if an error occurs or no metrics or dimensions
 are found
     */
    public CompletableFuture<Dimension> getSpecificMetAsync(String namespace) {
        ListMetricsRequest request = ListMetricsRequest.builder()
            .namespace(namespace)
            .build();

        return getAsyncClient().listMetrics(request).handle((response, exception) ->
 {
            if (exception != null) {
                logger.info("Error occurred while listing metrics: {} ",
 exception.getMessage());
                throw new RuntimeException("Failed to retrieve specific metric
 dimension", exception);
            } else {
                List<Metric> myList = response.metrics();
                if (!myList.isEmpty()) {
                    Metric metric = myList.get(0);
                    if (!metric.dimensions().isEmpty()) {
                        return metric.dimensions().get(0); // Return the first
 dimension
                    }
                }
                throw new RuntimeException("No metrics or dimensions found");
            }
        });
    }

    public static String readFileAsString(String file) throws IOException {
        return new String(Files.readAllBytes(Paths.get(file)));
    }
}
```

- 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

Azioni

`DeleteAlarms`

Il seguente esempio di codice mostra come usare. `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.

```
/**
 * Deletes a CloudWatch alarm.
 *
 * @param alarmName the name of the alarm to be deleted
```

```
    * @return a {@link CompletableFuture} representing the asynchronous operation
to delete the alarm
    * the {@link DeleteAlarmsResponse} is returned when the operation completes
successfully,
    * or a {@link RuntimeException} is thrown if the operation fails
    */
   public CompletableFuture<DeleteAlarmsResponse> deleteCWAlarmAsync(String
alarmName) {
       DeleteAlarmsRequest request = DeleteAlarmsRequest.builder()
           .alarmNames(alarmName)
           .build();

       return getAsyncClient().deleteAlarms(request)
           .whenComplete((response, exception) -> {
               if (exception != null) {
                   throw new RuntimeException("Failed to delete the alarm:{} " +
alarmName, exception);
               } else {
                   logger.info("Successfully deleted alarm {} ", alarmName);
               }
           });
   }
```

- Per i dettagli sull'API, consulta la DeleteAlarmssezione 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.

```
   /**
    * Deletes an Anomaly Detector.
    *
```

```
    * @param fileName the name of the file containing the Anomaly Detector
configuration
    * @return a CompletableFuture that represents the asynchronous deletion of the
Anomaly Detector
    */
   public CompletableFuture<DeleteAnomalyDetectorResponse>
deleteAnomalyDetectorAsync(String fileName) {
       CompletableFuture<JsonNode> readFileFuture =
CompletableFuture.supplyAsync(() -> {
           try {
               JsonParser parser = new JsonFactory().createParser(new
File(fileName));
               return new ObjectMapper().readTree(parser); // Return the root node
           } catch (IOException e) {
               throw new RuntimeException("Failed to read or parse the file", e);
           }
       });

       return readFileFuture.thenCompose(rootNode -> {
           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();

           return getAsyncClient().deleteAnomalyDetector(request);
       }).whenComplete((result, exception) -> {
           if (exception != null) {
               throw new RuntimeException("Failed to delete the Anomaly Detector",
exception);
           } else {
               logger.info("Successfully deleted the Anomaly Detector.");
           }
```

```
        });
    }
```

- Per i dettagli sull'API, consulta la DeleteAnomalyDetectorsezione 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.

```
    /**
     * Deletes the specified dashboard.
     *
     * @param dashboardName the name of the dashboard to be deleted
     * @return a {@link CompletableFuture} representing the asynchronous operation
of deleting the dashboard
     * @throws RuntimeException if the dashboard deletion fails
     */
    public CompletableFuture<DeleteDashboardsResponse> deleteDashboardAsync(String
dashboardName) {
        DeleteDashboardsRequest dashboardsRequest =
DeleteDashboardsRequest.builder()
            .dashboardNames(dashboardName)
            .build();

        return getAsyncClient().deleteDashboards(dashboardsRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    throw new RuntimeException("Failed to delete the dashboard: " +
dashboardName, exception);
                } else {
                    logger.info("{} was successfully deleted.", dashboardName);
```

```
                }
            });
    }
```

- Per i dettagli sull'API, consulta la DeleteDashboardssezione 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.

```
    /**
     * Retrieves the alarm history for a given alarm name and date range.
     *
     * @param fileName the path to the JSON file containing the alarm name
     * @param date     the date to start the alarm history search (in the format
"yyyy-MM-dd'T'HH:mm:ss'Z'")
     * @return a {@code CompletableFuture<Void>} that completes when the alarm
history has been retrieved and processed
     */
    public CompletableFuture<Void> getAlarmHistoryAsync(String fileName, String
date) {
        CompletableFuture<String> readFileFuture = CompletableFuture.supplyAsync(()
-> {
            try {
                JsonParser parser = new JsonFactory().createParser(new
File(fileName));
                com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
                return rootNode.findValue("exampleAlarmName").asText(); // Return
alarmName from the JSON file
```

```
            } catch (IOException e) {
                throw new RuntimeException("Failed to read or parse the file", e);
            }
        });

        // Use the alarm name to describe alarm history with a paginator.
        return readFileFuture.thenCompose(alarmName -> {
            try {
                Instant start = Instant.parse(date);
                Instant endDate = Instant.now();
                DescribeAlarmHistoryRequest historyRequest =
DescribeAlarmHistoryRequest.builder()
                    .startDate(start)
                    .endDate(endDate)
                    .alarmName(alarmName)
                    .historyItemType(HistoryItemType.ACTION)
                    .build();

                // Use the paginator to paginate through alarm history pages.
                DescribeAlarmHistoryPublisher historyPublisher =
getAsyncClient().describeAlarmHistoryPaginator(historyRequest);
                CompletableFuture<Void> future = historyPublisher
                    .subscribe(response -> response.alarmHistoryItems().forEach(item
-> {
                        logger.info("History summary: {}", item.historySummary());
                        logger.info("Timestamp: {}", item.timestamp());
                    }))
                    .whenComplete((result, exception) -> {
                        if (exception != null) {
                            logger.error("Error occurred while getting alarm
history: " + exception.getMessage(), exception);
                        } else {
                            logger.info("Successfully retrieved all alarm
history.");
                        }
                    });

                // Return the future to the calling code for further handling
                return future;
            } catch (Exception e) {
                throw new RuntimeException("Failed to process alarm history", e);
            }
        }).whenComplete((result, exception) -> {
            if (exception != null) {
```

```
                throw new RuntimeException("Error completing alarm history
processing", exception);
            }
        });
    }
```

- Per i dettagli sull'API, consulta la DescribeAlarmHistorysezione 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.

```
    /**
     * Describes the CloudWatch alarms of the 'METRIC_ALARM' type.
     *
     * @return a {@link CompletableFuture} that represents the asynchronous
operation
     * of describing the CloudWatch alarms. The future completes when the
     * operation is finished, either successfully or with an error.
     */
    public CompletableFuture<Void> describeAlarmsAsync() {
        List<AlarmType> typeList = new ArrayList<>();
        typeList.add(AlarmType.METRIC_ALARM);
        DescribeAlarmsRequest alarmsRequest = DescribeAlarmsRequest.builder()
            .alarmTypes(typeList)
            .maxRecords(10)
            .build();

        return getAsyncClient().describeAlarms(alarmsRequest)
```

```
            .thenAccept(response -> {
                List<MetricAlarm> alarmList = response.metricAlarms();
                for (MetricAlarm alarm : alarmList) {
                    logger.info("Alarm name: {}", alarm.alarmName());
                    logger.info("Alarm description: {} ", alarm.alarmDescription());
                }
            })
            .whenComplete((response, ex) -> {
                if (ex != null) {
                    logger.info("Failed to describe alarms: {}", ex.getMessage());
                } else {
                    logger.info("Successfully described alarms.");
                }
            });
    }
```

- Per i dettagli sull'API, consulta la DescribeAlarmssezione 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.

```
    /**
     * Checks for a metric alarm in AWS CloudWatch.
     *
     * @param fileName the name of the file containing the JSON configuration for
the custom metric
     * @return a {@link CompletableFuture} that completes when the check for the
metric alarm is complete
     */
```

```
   public CompletableFuture<Void> checkForMetricAlarmAsync(String fileName) {
       CompletableFuture<String> readFileFuture = CompletableFuture.supplyAsync(()
-> {
           try {
               JsonParser parser = new JsonFactory().createParser(new
File(fileName));
               com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
               return rootNode.toString(); // Return JSON as a string for further
processing
           } catch (IOException e) {
               throw new RuntimeException("Failed to read file", e);
           }
       });

       return readFileFuture.thenCompose(jsonContent -> {
           try {
               com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(jsonContent);
               String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
               String customMetricName =
rootNode.findValue("customMetricName").asText();

               DescribeAlarmsForMetricRequest metricRequest =
DescribeAlarmsForMetricRequest.builder()
                   .metricName(customMetricName)
                   .namespace(customMetricNamespace)
                   .build();

               return checkForAlarmAsync(metricRequest, customMetricName, 10);

           } catch (IOException e) {
               throw new RuntimeException("Failed to parse JSON content", e);
           }
       }).whenComplete((result, exception) -> {
           if (exception != null) {
               throw new RuntimeException("Error checking metric alarm",
exception);
           }
       });
   }

   // Recursive method to check for the alarm.
```

```
   /**
    * Checks for the existence of an alarm asynchronously for the specified metric.
    *
    * @param metricRequest    the request to describe the alarms for the specified
metric
    * @param customMetricName the name of the custom metric to check for an alarm
    * @param retries          the number of retries to perform if no alarm is found
    * @return a {@link CompletableFuture} that completes when an alarm is found or
the maximum number of retries has been reached
    */
   private static CompletableFuture<Void>
checkForAlarmAsync(DescribeAlarmsForMetricRequest metricRequest, String
customMetricName, int retries) {
       if (retries == 0) {
           return CompletableFuture.completedFuture(null).thenRun(() ->
               logger.info("No Alarm state found for {} after 10 retries.",
customMetricName)
           );
       }

       return
(getAsyncClient().describeAlarmsForMetric(metricRequest).thenCompose(response -> {
           if (response.hasMetricAlarms()) {
               logger.info("Alarm state found for {}", customMetricName);
               return CompletableFuture.completedFuture(null); // Alarm found,
complete the future
           } else {
               return CompletableFuture.runAsync(() -> {
                   try {
                       Thread.sleep(20000);
                       logger.info(".");
                   } catch (InterruptedException e) {
                       throw new RuntimeException("Interrupted while waiting to
retry", e);
                   }
               }).thenCompose(v -> checkForAlarmAsync(metricRequest,
customMetricName, retries - 1)); // Recursive call
           }
       }));
   }
```

- Per i dettagli sull'API, consulta la DescribeAlarmsForMetricsezione 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.

```
   /**
    * Describes the anomaly detectors based on the specified JSON file.
    *
    * @param fileName the name of the JSON file containing the custom metric
namespace and name
    * @return a {@link CompletableFuture} that completes when the anomaly detectors
have been described
    * @throws RuntimeException if there is a failure during the operation, such as
when reading or parsing the JSON file,
    *                          or when describing the anomaly detectors
    */
   public CompletableFuture<Void> describeAnomalyDetectorsAsync(String fileName) {
       CompletableFuture<JsonNode> readFileFuture =
CompletableFuture.supplyAsync(() -> {
           try {
               JsonParser parser = new JsonFactory().createParser(new
File(fileName));
               return new ObjectMapper().readTree(parser);
           } catch (IOException e) {
               throw new RuntimeException("Failed to read or parse the file", e);
           }
       });

       return readFileFuture.thenCompose(rootNode -> {
           try {
```

```
                String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
                String customMetricName =
rootNode.findValue("customMetricName").asText();

                DescribeAnomalyDetectorsRequest detectorsRequest =
DescribeAnomalyDetectorsRequest.builder()
                    .maxResults(10)
                    .metricName(customMetricName)
                    .namespace(customMetricNamespace)
                    .build();

                return
getAsyncClient().describeAnomalyDetectors(detectorsRequest).thenAccept(response ->
{
                    List<AnomalyDetector> anomalyDetectorList =
response.anomalyDetectors();
                    for (AnomalyDetector detector : anomalyDetectorList) {
                        logger.info("Metric name: {} ",
detector.singleMetricAnomalyDetector().metricName());
                        logger.info("State: {} ", detector.stateValue());
                    }
                });
            } catch (RuntimeException e) {
                throw new RuntimeException("Failed to describe anomaly detectors",
e);
            }
        }).whenComplete((result, exception) -> {
            if (exception != null) {
                throw new RuntimeException("Error describing anomaly detectors",
exception);
            }
        });
    }
```

- Per i dettagli sull'API, consulta la DescribeAnomalyDetectorssezione 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 DisableAlarmActionssezione 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 EnableAlarmActionssezione 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.

```
    /**
     * Retrieves custom metric data from the AWS CloudWatch service.
     *
     * @param fileName the name of the file containing the custom metric information
     * @return a {@link CompletableFuture} that completes when the metric data has
been retrieved
     */
    public CompletableFuture<Void> getCustomMetricDataAsync(String fileName) {
        CompletableFuture<String> readFileFuture = CompletableFuture.supplyAsync(()
-> {
            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);
                return rootNode.toString(); // Return JSON as a string for further
processing
            } catch (IOException e) {
                throw new RuntimeException("Failed to read file", e);
            }
        });

        return readFileFuture.thenCompose(jsonContent -> {
            try {
                // Parse the JSON string to extract relevant values.
                com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(jsonContent);
                String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
                String customMetricName =
rootNode.findValue("customMetricName").asText();

                // Set the current time and date range for metric query.
                Instant nowDate = Instant.now();
                long hours = 1;
                long minutes = 30;
                Instant endTime = 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(60)  // Assuming period in seconds
                    .metric(met)
                    .build();

                MetricDataQuery dataQuery = MetricDataQuery.builder()
                    .metricStat(metStat)
                    .id("foo2")
                    .returnData(true)
                    .build();

                List<MetricDataQuery> dq = new ArrayList<>();
```

```
                dq.add(dataQuery);

                GetMetricDataRequest getMetricDataRequest =
GetMetricDataRequest.builder()
                    .maxDatapoints(10)
                    .scanBy(ScanBy.TIMESTAMP_DESCENDING)
                    .startTime(nowDate)
                    .endTime(endTime)
                    .metricDataQueries(dq)
                    .build();

                // Call the async method for CloudWatch data retrieval.
                return getAsyncClient().getMetricData(getMetricDataRequest);

            } catch (IOException e) {
                throw new RuntimeException("Failed to parse JSON content", e);
            }
        }).thenAccept(response -> {
            List<MetricDataResult> data = response.metricDataResults();
            for (MetricDataResult item : data) {
                logger.info("The label is: {}", item.label());
                logger.info("The status code is: {}", item.statusCode().toString());
            }
        }).exceptionally(exception -> {
            throw new RuntimeException("Failed to get metric data", exception);
        });
    }
```

- Per i dettagli sull'API, consulta la GetMetricDatasezione 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.

```
   /**
    * Retrieves and displays metric statistics for the specified parameters.
    *
    * @param nameSpace    the namespace for the metric
    * @param metVal       the name of the metric
    * @param metricOption the statistic to retrieve for the metric (e.g.,
"Maximum", "Average")
    * @param date         the date for which to retrieve the metric statistics, in
the format "yyyy-MM-dd'T'HH:mm:ss'Z'"
    * @param myDimension  the dimension(s) to filter the metric statistics by
    * @return a {@link CompletableFuture} that completes when the metric statistics
have been retrieved and displayed
    */
   public CompletableFuture<GetMetricStatisticsResponse>
getAndDisplayMetricStatisticsAsync(String nameSpace, String metVal,

        String metricOption, String date, Dimension myDimension) {

       Instant start = Instant.parse(date);
       Instant endDate = Instant.now();

       // Building the request for metric statistics.
       GetMetricStatisticsRequest statisticsRequest =
GetMetricStatisticsRequest.builder()
           .endTime(endDate)
           .startTime(start)
           .dimensions(myDimension)
           .metricName(metVal)
           .namespace(nameSpace)
           .period(86400) // 1 day period
           .statistics(Statistic.fromValue(metricOption))
           .build();

       return getAsyncClient().getMetricStatistics(statisticsRequest)
           .whenComplete((response, exception) -> {
               if (response != null) {
                   List<Datapoint> data = response.datapoints();
                   if (!data.isEmpty()) {
                       for (Datapoint datapoint : data) {
                           logger.info("Timestamp: {} Maximum value: {}",
datapoint.timestamp(), datapoint.maximum());
                       }
                   } else {
```

```
                    logger.info("The returned data list is empty");
                }
            } else {
                logger.info("Failed to get metric statistics: {} ",
exception.getMessage());
            }
        })
        .exceptionally(exception -> {
            throw new RuntimeException("Error while getting metric statistics: "
+ exception.getMessage(), exception);
        });
   }
```

- Per i dettagli sull'API, consulta la GetMetricStatisticssezione 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.

```
   /**
    * Retrieves and saves a custom metric image to a file.
    *
    * @param fileName the name of the file to save the metric image to
    * @return a {@link CompletableFuture} that completes when the image has been
saved to the file
    */
   public CompletableFuture<Void> downloadAndSaveMetricImageAsync(String fileName)
{
       logger.info("Getting Image data for custom metric.");
```

```
        String myJSON = """
              {
                  "title": "Example Metric Graph",
                  "view": "timeSeries",
                  "stacked ": false,
                  "period": 10,
                  "width": 1400,
                  "height": 600,
                  "metrics": [
                      [
                      "AWS/Billing",
                      "EstimatedCharges",
                      "Currency",
                      "USD"
                     ]
                  ]
              }
             """;

        GetMetricWidgetImageRequest imageRequest =
GetMetricWidgetImageRequest.builder()
            .metricWidget(myJSON)
            .build();

        return getAsyncClient().getMetricWidgetImage(imageRequest)
            .thenCompose(response -> {
                SdkBytes sdkBytes = response.metricWidgetImage();
                byte[] bytes = sdkBytes.asByteArray();
                return CompletableFuture.runAsync(() -> {
                    try {
                        File outputFile = new File(fileName);
                        try (FileOutputStream outputStream = new
FileOutputStream(outputFile)) {
                            outputStream.write(bytes);
                        }
                    } catch (IOException e) {
                        throw new RuntimeException("Failed to write image to file",
e);
                    }
                });
            })
            .whenComplete((result, exception) -> {
                if (exception != null) {
```

```
                    throw new RuntimeException("Error getting and saving metric
image", exception);
                } else {
                    logger.info("Image data saved successfully to {}", fileName);
                }
            });
    }
```

- Per i dettagli sull'API, consulta la GetMetricWidgetImagesezione 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.

```
    /**
     * Lists the available dashboards.
     *
     * @return a {@link CompletableFuture} that completes when the operation is
finished.
     * The future will complete exceptionally if an error occurs while listing the
dashboards.
     */
    public CompletableFuture<Void> listDashboardsAsync() {
        ListDashboardsRequest listDashboardsRequest =
ListDashboardsRequest.builder().build();
        ListDashboardsPublisher paginator =
getAsyncClient().listDashboardsPaginator(listDashboardsRequest);
        return paginator.subscribe(response -> {
            response.dashboardEntries().forEach(entry -> {
                logger.info("Dashboard name is: {} ", entry.dashboardName());
                logger.info("Dashboard ARN is: {} ", entry.dashboardArn());
```

```
            });
        }).exceptionally(ex -> {
            logger.info("Failed to list dashboards: {} ", ex.getMessage());
            throw new RuntimeException("Error occurred while listing dashboards",
ex);
        });
    }
```

- Per i dettagli sull'API, consulta la ListDashboardssezione 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.

```
    /**
     * Retrieves a list of metric names for the specified namespace.
     *
     * @param namespace the namespace for which to retrieve the metric names
     * @return a {@link CompletableFuture} that, when completed, contains an {@link
ArrayList} of
     * the metric names in the specified namespace
     * @throws RuntimeException if an error occurs while listing the metrics
     */
    public CompletableFuture<ArrayList<String>> listMetsAsync(String namespace) {
        ListMetricsRequest request = ListMetricsRequest.builder()
            .namespace(namespace)
            .build();

        ListMetricsPublisher metricsPaginator =
getAsyncClient().listMetricsPaginator(request);
        Set<String> metSet = new HashSet<>();
```

```
        CompletableFuture<Void> future = metricsPaginator.subscribe(response -> {
            response.metrics().forEach(metric -> {
                String metricName = metric.metricName();
                metSet.add(metricName);
            });
        });

        return future
            .thenApply(ignored -> new ArrayList<>(metSet))
            .exceptionally(exception -> {
                throw new RuntimeException("Failed to list metrics: " +
exception.getMessage(), exception);
            });
    }
```

- Per i dettagli sull'API, consulta la ListMetricssezione 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.

```
    /**
     * Adds an anomaly detector for the given file.
     *
     * @param fileName the name of the file containing the anomaly detector
configuration
     * @return a {@link CompletableFuture} that completes when the anomaly detector
has been added
     */
    public CompletableFuture<Void> addAnomalyDetectorAsync(String fileName) {
        CompletableFuture<JsonNode> readFileFuture =
CompletableFuture.supplyAsync(() -> {
```

```
            try {
                JsonParser parser = new JsonFactory().createParser(new
File(fileName));
                return new ObjectMapper().readTree(parser); // Return the root node
            } catch (IOException e) {
                throw new RuntimeException("Failed to read or parse the file", e);
            }
        });

        return readFileFuture.thenCompose(rootNode -> {
            try {
                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();

                return
getAsyncClient().putAnomalyDetector(anomalyDetectorRequest).thenAccept(response ->
{
                    logger.info("Added anomaly detector for metric {}",
customMetricName);
                });
            } catch (Exception e) {
                throw new RuntimeException("Failed to create anomaly detector", e);
            }
        }).whenComplete((result, exception) -> {
            if (exception != null) {
                throw new RuntimeException("Error adding anomaly detector",
exception);
            }
        });
    }
```

- Per i dettagli sull'API, consulta la PutAnomalyDetectorsezione 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.

```
   /**
    * Creates a new dashboard with the specified name and metrics from the given
file.
    *
    * @param dashboardName the name of the dashboard to be created
    * @param fileName      the name of the file containing the dashboard body
    * @return a {@link CompletableFuture} representing the asynchronous operation
of creating the dashboard
    * @throws IOException if there is an error reading the dashboard body from the
file
    */
   public CompletableFuture<PutDashboardResponse>
createDashboardWithMetricsAsync(String dashboardName, String fileName) throws
IOException {
       String dashboardBody = readFileAsString(fileName);
       PutDashboardRequest dashboardRequest = PutDashboardRequest.builder()
           .dashboardName(dashboardName)
           .dashboardBody(dashboardBody)
           .build();

       return getAsyncClient().putDashboard(dashboardRequest)
           .handle((response, ex) -> {
               if (ex != null) {
```

```
                    logger.info("Failed to create dashboard: {}", ex.getMessage());
                    throw new RuntimeException("Dashboard creation failed", ex);
                } else {
                    // Handle the normal response case
                    logger.info("{} was successfully created.", dashboardName);
                    List<DashboardValidationMessage> messages =
response.dashboardValidationMessages();
                    if (messages.isEmpty()) {
                        logger.info("There are no messages in the new Dashboard.");
                    } else {
                        for (DashboardValidationMessage message : messages) {
                            logger.info("Message: {}", message.message());
                        }
                    }
                    return response; // Return the response for further use
                }
            });
    }
```

- Per i dettagli sull'API, consulta la PutDashboardsezione 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.

```
    /**
     * Creates an alarm based on the configuration provided in a JSON file.
     *
     * @param fileName the name of the JSON file containing the alarm configuration
     * @return a CompletableFuture that represents the asynchronous operation of
creating the alarm
     * @throws RuntimeException if an exception occurs while reading the JSON file
or creating the alarm
```

```
     */
    public CompletableFuture<String> createAlarmAsync(String fileName) {
        com.fasterxml.jackson.databind.JsonNode rootNode;
        try {
            JsonParser parser = new JsonFactory().createParser(new File(fileName));
            rootNode = new ObjectMapper().readTree(parser);
        } catch (IOException e) {
            throw new RuntimeException("Failed to read the alarm configuration
file", e);
        }

        // Extract values from the JSON node.
        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();

        // Call the putMetricAlarm asynchronously and handle the result.
        return getAsyncClient().putMetricAlarm(alarmRequest)
            .handle((response, ex) -> {
```

```
                if (ex != null) {
                    logger.info("Failed to create alarm: {}", ex.getMessage());
                    throw new RuntimeException("Failed to create alarm", ex);
                } else {
                    logger.info("{} was successfully created!", alarmName);
                    return alarmName;
                }
            });
    }
```

- Per i dettagli sull'API, consulta la PutMetricAlarmsezione 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.

```
    /**
     * Adds metric data for an alarm asynchronously.
     *
     * @param fileName the name of the JSON file containing the metric data
     * @return a CompletableFuture that asynchronously returns the
PutMetricDataResponse
     */
    public CompletableFuture<PutMetricDataResponse>
addMetricDataForAlarmAsync(String fileName) {
        CompletableFuture<String> readFileFuture = CompletableFuture.supplyAsync(()
-> {
            try {
                JsonParser parser = new JsonFactory().createParser(new
File(fileName));
                com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
```

```
                return rootNode.toString(); // Return JSON as a string for further
processing
            } catch (IOException e) {
                throw new RuntimeException("Failed to read file", e);
            }
        });

        return readFileFuture.thenCompose(jsonContent -> {
            try {
                com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(jsonContent);
                String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
                String customMetricName =
rootNode.findValue("customMetricName").asText();
                Instant instant = Instant.now();

                // Create MetricDatum objects.
                MetricDatum datum1 = 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(datum1);
                metricDataList.add(datum2);

                // Build the PutMetricData request.
                PutMetricDataRequest request = PutMetricDataRequest.builder()
                    .namespace(customMetricNamespace)
                    .metricData(metricDataList)
                    .build();

                // Send the request asynchronously.
                return getAsyncClient().putMetricData(request);
```

```
            } catch (IOException e) {
                CompletableFuture<PutMetricDataResponse> failedFuture = new
CompletableFuture<>();
                failedFuture.completeExceptionally(new RuntimeException("Failed to
parse JSON content", e));
                return failedFuture;
            }
        }).whenComplete((response, exception) -> {
            if (exception != null) {
                logger.error("Failed to put metric data: " + exception.getMessage(),
exception);
            } else {
                logger.info("Added metric values for metric.");
            }
        });
    }
```

- Per i dettagli sull'API, consulta la PutMetricDatasezione AWS SDK for Java 2.x API Reference.

Scenari

Monitora le prestazioni di DynamoDB

Il seguente esempio di codice mostra come configurare l'uso di DynamoDB da parte di un'applicazione per monitorare le prestazioni.

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

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.

Ogni esempio include un collegamento al codice sorgente completo, in cui è possibile 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.

```
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatchevents.CloudWatchEventsClient;
import software.amazon.awssdk.services.cloudwatchevents.model.PutEventsRequest;
import software.amazon.awssdk.services.cloudwatchevents.model.PutEventsRequestEntry;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
```

```
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class PutEvents {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                   <resourceArn>

                Where:
                   resourceArn - An Amazon Resource Name (ARN) related to the
 events.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String resourceArn = args[0];
        CloudWatchEventsClient cwe = CloudWatchEventsClient.builder()
                .build();

        putCWEvents(cwe, resourceArn);
        cwe.close();
    }

    public static void putCWEvents(CloudWatchEventsClient cwe, String resourceArn) {
        try {
            final String EVENT_DETAILS = "{ \"key1\": \"value1\", \"key2\":
 \"value2\" }";

            PutEventsRequestEntry requestEntry = PutEventsRequestEntry.builder()
                    .detail(EVENT_DETAILS)
                    .detailType("sampleSubmitted")
                    .resources(resourceArn)
                    .source("aws-sdk-java-cloudwatch-example")
                    .build();

            PutEventsRequest request = PutEventsRequest.builder()
                    .entries(requestEntry)
                    .build();

            cwe.putEvents(request);
```

```
            System.out.println("Successfully put CloudWatch event");

        } catch (CloudWatchException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Per i dettagli sull'API, consulta la PutEventssezione 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",
                    roleArn, response.ruleArn());

        } catch (CloudWatchException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Per i dettagli sull'API, consulta la PutRulesezione 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 PutTargetssezione 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.

Gli scenari sono esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio o combinate con altri Servizi AWS.

Ogni esempio include un collegamento al codice sorgente completo, in cui è possibile trovare istruzioni su come configurare ed eseguire il codice nel contesto.

Argomenti

- Azioni
- Scenari

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 DeleteSubscriptionFiltersezione 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 DescribeSubscriptionFilterssezione 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 cwl = CloudWatchLogsClient.builder()
                .region(region)
                .build();

        putSubFilters(cwl, filter, pattern, logGroup, functionArn);
        cwl.close();
    }

    public static void putSubFilters(CloudWatchLogsClient cwl,
            String filter,
            String pattern,
            String logGroup,
            String functionArn) {

        try {
            PutSubscriptionFilterRequest request =
PutSubscriptionFilterRequest.builder()
                    .filterName(filter)
                    .filterPattern(pattern)
                    .logGroupName(logGroup)
                    .destinationArn(functionArn)
                    .build();
```

```
            cwl.putSubscriptionFilter(request);
            System.out.printf(
                    "Successfully created CloudWatch logs subscription filter %s",
                    filter);

        } catch (CloudWatchLogsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Per i dettagli sull'API, consulta la PutSubscriptionFiltersezione 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.

Scenari

Utilizzo degli eventi pianificati per richiamare una funzione Lambda

Il seguente esempio di codice mostra come creare una AWS Lambda funzione richiamata da un evento EventBridge pianificato di Amazon.

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

- CloudWatch Registri
- DynamoDB
- EventBridge
- Lambda
- Amazon SNS

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.

Ogni esempio include un collegamento al codice sorgente completo, 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.CognitoIdentityProviderExcept

/**
 * Before running this Java V2 code example, set up your development
```

```
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class 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 CreateIdentityPoolsezione 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 identityPoold = args[0];
        CognitoIdentityClient cognitoIdClient = CognitoIdentityClient.builder()
                .region(Region.US_EAST_1)
                .credentialsProvider(ProfileCredentialsProvider.create())
                .build();

        deleteIdPool(cognitoIdClient, identityPoold);
        cognitoIdClient.close();
    }

    public static void deleteIdPool(CognitoIdentityClient cognitoIdClient, String
 identityPoold) {
        try {

            DeleteIdentityPoolRequest identityPoolRequest =
 DeleteIdentityPoolRequest.builder()
                    .identityPoolId(identityPoold)
                    .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 DeleteIdentityPoolsezione 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.CognitoIdentityProviderExcept

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class GetIdentityCredentials {
    public static void main(String[] args) {

        final String usage = """

                Usage:
                    <identityId>\s
```

```
            Where:
                identityId - The Id of an existing identity in the format
 REGION:GUID.
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String identityId = args[0];
        CognitoIdentityClient cognitoClient = CognitoIdentityClient.builder()
                .region(Region.US_EAST_1)
                .build();

        getCredsForIdentity(cognitoClient, identityId);
        cognitoClient.close();
    }

    public static void getCredsForIdentity(CognitoIdentityClient cognitoClient,
 String identityId) {
        try {
            GetCredentialsForIdentityRequest getCredentialsForIdentityRequest =
 GetCredentialsForIdentityRequest
                    .builder()
                    .identityId(identityId)
                    .build();

            GetCredentialsForIdentityResponse response = cognitoClient
                    .getCredentialsForIdentity(getCredentialsForIdentityRequest);
            System.out.println(
                    "Identity ID " + response.identityId() + ", Access key ID " +
 response.credentials().accessKeyId());

        } catch (CognitoIdentityProviderException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Per i dettagli sull'API, consulta la GetCredentialsForIdentitysezione 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.CognitoIdentityProviderExcept

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListIdentityPools {
    public static void main(String[] args) {
        CognitoIdentityClient cognitoClient = CognitoIdentityClient.builder()
                .region(Region.US_EAST_1)
                .build();

        listIdPools(cognitoClient);
        cognitoClient.close();
    }
```

```
    public static void listIdPools(CognitoIdentityClient cognitoClient) {
        try {
            ListIdentityPoolsRequest poolsRequest =
 ListIdentityPoolsRequest.builder()
                    .maxResults(15)
                    .build();

            ListIdentityPoolsResponse response =
 cognitoClient.listIdentityPools(poolsRequest);
            response.identityPools().forEach(pool -> {
                System.out.println("Pool ID: " + pool.identityPoolId());
                System.out.println("Pool name: " + pool.identityPoolName());
            });

        } catch (CognitoIdentityProviderException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Per i dettagli sull'API, consulta la ListIdentityPoolssezione 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.

Gli scenari sono esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio o combinate con altri Servizi AWS.

Ogni esempio include un collegamento al codice sorgente completo, 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.CognitoIdentityProviderExcept
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 ListUserPoolssezione 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 AdminGetUsersezione 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 AdminInitiateAuthsezione 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 AdminRespondToAuthChallengesezione 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 AssociateSoftwareTokensezione 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 ConfirmSignUpsezione 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.CognitoIdentityProviderExcept
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 CreateUserPoolsezione 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.CognitoIdentityProviderExcept
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, consulta la CreateUserPoolClientsezione 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.CognitoIdentityProviderExcept
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 ListUserPoolssezione 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.CognitoIdentityProviderExcept
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, consulta la ListUserssezione 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, consulta la ResendConfirmationCodesezione 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, consulta la SignUpsezione 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, consulta la VerifySoftwareTokensezione 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.AdminRespondToAuthChallengeRe
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.AdminRespondToAuthChallengeRe
```

```
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.AssociateSoftwareTokenRequest
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.AssociateSoftwareTokenRespons
import software.amazon.awssdk.services.cognitoidentityprovider.model.AttributeType;
import software.amazon.awssdk.services.cognitoidentityprovider.model.AuthFlowType;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.ChallengeNameType;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderExcept
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.ConfirmSignUpRequest;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.ResendConfirmationCodeRequest
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.ResendConfirmationCodeRespons
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.

Gli scenari sono esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio o combinate con altri Servizi AWS.

Ogni esempio include un collegamento al codice sorgente completo, dove puoi trovare istruzioni su come configurare ed eseguire il codice nel contesto.

Argomenti

- Azioni
- Scenari

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, consulta la CreateDocumentClassifiersezione 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, consulta la DetectDominantLanguagesezione 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, consulta la DetectEntitiessezione 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, consulta la DetectKeyPhrasessezione 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, consulta la DetectSentimentsezione 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, consulta la DetectSyntaxsezione AWS SDK for Java 2.x API Reference.

Scenari

Creazione di un chatbot Amazon Lex

Il seguente esempio di codice mostra come creare un chatbot 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

Crea un'applicazione di messaggistica

Il seguente esempio di codice mostra come creare un'applicazione di messaggistica 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

Crea un'applicazione per analizzare il feedback dei clienti

L'esempio di codice seguente mostra come creare un'applicazione che analizza le schede dei commenti dei clienti, le traduce dalla loro lingua originale, ne determina il sentiment e genera un file audio dal testo tradotto.

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

Esempi di Firehose 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 Firehose.

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.

Gli scenari sono esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio o combinate con altri Servizi AWS.

Ogni esempio include un collegamento al codice sorgente completo, in cui è possibile trovare istruzioni su come configurare ed eseguire il codice nel contesto.

Argomenti

- Azioni
- Scenari

Azioni

`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.

```
   /**
    * Puts a record to the specified Amazon Kinesis Data Firehose delivery stream.
    *
    * @param record The record to be put to the delivery stream. The record must be
a {@link Map} of String keys and Object values.
    * @param deliveryStreamName The name of the Amazon Kinesis Data Firehose
delivery stream to which the record should be put.
    * @throws IllegalArgumentException if the input record or delivery stream name
is null or empty.
    * @throws RuntimeException if there is an error putting the record to the
delivery stream.
    */
   public static void putRecord(Map<String, Object> record, String
deliveryStreamName) {
       if (record == null || deliveryStreamName == null ||
deliveryStreamName.isEmpty()) {
           throw new IllegalArgumentException("Invalid input: record or delivery
stream name cannot be null/empty");
       }
       try {
           String jsonRecord = new ObjectMapper().writeValueAsString(record);
           Record firehoseRecord = Record.builder()

.data(SdkBytes.fromByteArray(jsonRecord.getBytes(StandardCharsets.UTF_8)))
               .build();

           PutRecordRequest putRecordRequest = PutRecordRequest.builder()
               .deliveryStreamName(deliveryStreamName)
```

```
                .record(firehoseRecord)
                .build();

            getFirehoseClient().putRecord(putRecordRequest);
            System.out.println("Record sent: " + jsonRecord);
        } catch (Exception e) {
            throw new RuntimeException("Failed to put record: " + e.getMessage(),
e);
        }
    }
```

- Per i dettagli sull'API, consulta la PutRecordsezione AWS SDK for Java 2.x API Reference.

`PutRecordBatch`

Il seguente esempio di codice mostra come utilizzare`PutRecordBatch`.

SDK per Java 2.x

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
    /**
     * Puts a batch of records to an Amazon Kinesis Data Firehose delivery stream.
     *
     * @param records           a list of maps representing the records to be sent
     * @param batchSize         the maximum number of records to include in each
batch
     * @param deliveryStreamName the name of the Kinesis Data Firehose delivery
stream
     * @throws IllegalArgumentException if the input parameters are invalid (null or
empty)
     * @throws RuntimeException         if there is an error putting the record
batch
     */
    public static void putRecordBatch(List<Map<String, Object>> records, int
batchSize, String deliveryStreamName) {
```

```
        if (records == null || records.isEmpty() || deliveryStreamName == null ||
deliveryStreamName.isEmpty()) {
            throw new IllegalArgumentException("Invalid input: records or delivery
stream name cannot be null/empty");
        }
        ObjectMapper objectMapper = new ObjectMapper();

        try {
            for (int i = 0; i < records.size(); i += batchSize) {
                List<Map<String, Object>> batch = records.subList(i, Math.min(i +
batchSize, records.size()));

                List<Record> batchRecords = batch.stream().map(record -> {
                    try {
                        String jsonRecord = objectMapper.writeValueAsString(record);
                        return Record.builder()

.data(SdkBytes.fromByteArray(jsonRecord.getBytes(StandardCharsets.UTF_8)))
                                .build();
                    } catch (Exception e) {
                        throw new RuntimeException("Error creating Firehose record",
e);
                    }
                }).collect(Collectors.toList());

                PutRecordBatchRequest request = PutRecordBatchRequest.builder()
                    .deliveryStreamName(deliveryStreamName)
                    .records(batchRecords)
                    .build();

                PutRecordBatchResponse response =
getFirehoseClient().putRecordBatch(request);

                if (response.failedPutCount() > 0) {
                    response.requestResponses().stream()
                        .filter(r -> r.errorCode() != null)
                        .forEach(r -> System.err.println("Failed record: " +
r.errorMessage()));
                }
                System.out.println("Batch sent with size: " + batchRecords.size());
            }
        } catch (Exception e) {
            throw new RuntimeException("Failed to put record batch: " +
e.getMessage(), e);
```

```
        }
    }
```

- Per i dettagli sull'API, consulta la PutRecordBatchsezione AWS SDK for Java 2.x API Reference.

Scenari

Mettere i record su Firehose

Il seguente esempio di codice mostra come utilizzare Firehose per elaborare record singoli e in batch.

SDK per Java 2.x

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

Questo esempio inserisce record singoli e batch in Firehose.

```
/**
 * Amazon Firehose Scenario example using Java V2 SDK.
 *
 * Demonstrates individual and batch record processing,
 * and monitoring Firehose delivery stream metrics.
 */
public class FirehoseScenario {

    private static FirehoseClient firehoseClient;
    private static CloudWatchClient cloudWatchClient;

    public static void main(String[] args) {
        final String usage = """
                Usage:
                    <deliveryStreamName>
                Where:
                    deliveryStreamName - The Firehose delivery stream name.
                """;
```

```
        if (args.length != 1) {
            System.out.println(usage);
            return;
        }

        String deliveryStreamName = args[0];

        try {
            // Read and parse sample data.
            String jsonContent = readJsonFile("sample_records.json");
            ObjectMapper objectMapper = new ObjectMapper();
            List<Map<String, Object>> sampleData =
objectMapper.readValue(jsonContent, new TypeReference<>() {});

            // Process individual records.
            System.out.println("Processing individual records...");
            sampleData.subList(0, 100).forEach(record -> {
                try {
                    putRecord(record, deliveryStreamName);
                } catch (Exception e) {
                    System.err.println("Error processing record: " +
e.getMessage());
                }
            });

            // Monitor metrics.
            monitorMetrics(deliveryStreamName);

            // Process batch records.
            System.out.println("Processing batch records...");
            putRecordBatch(sampleData.subList(100, sampleData.size()), 500,
deliveryStreamName);
            monitorMetrics(deliveryStreamName);

        } catch (Exception e) {
            System.err.println("Scenario failed: " + e.getMessage());
        } finally {
            closeClients();
        }
    }

    private static FirehoseClient getFirehoseClient() {
        if (firehoseClient == null) {
            firehoseClient = FirehoseClient.builder()
```

```
                    .region(Region.US_EAST_1)
                    .build();
        }
        return firehoseClient;
    }

    private static CloudWatchClient getCloudWatchClient() {
        if (cloudWatchClient == null) {
            cloudWatchClient = CloudWatchClient.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }
        return cloudWatchClient;
    }

    /**
     * Puts a record to the specified Amazon Kinesis Data Firehose delivery stream.
     *
     * @param record The record to be put to the delivery stream. The record must be
a {@link Map} of String keys and Object values.
     * @param deliveryStreamName The name of the Amazon Kinesis Data Firehose
delivery stream to which the record should be put.
     * @throws IllegalArgumentException if the input record or delivery stream name
is null or empty.
     * @throws RuntimeException if there is an error putting the record to the
delivery stream.
     */
    public static void putRecord(Map<String, Object> record, String
deliveryStreamName) {
        if (record == null || deliveryStreamName == null ||
deliveryStreamName.isEmpty()) {
            throw new IllegalArgumentException("Invalid input: record or delivery
stream name cannot be null/empty");
        }
        try {
            String jsonRecord = new ObjectMapper().writeValueAsString(record);
            Record firehoseRecord = Record.builder()

.data(SdkBytes.fromByteArray(jsonRecord.getBytes(StandardCharsets.UTF_8)))
                .build();

            PutRecordRequest putRecordRequest = PutRecordRequest.builder()
                .deliveryStreamName(deliveryStreamName)
                .record(firehoseRecord)
```

```
                .build();

            getFirehoseClient().putRecord(putRecordRequest);
            System.out.println("Record sent: " + jsonRecord);
        } catch (Exception e) {
            throw new RuntimeException("Failed to put record: " + e.getMessage(),
e);
        }
    }


    /**
     * Puts a batch of records to an Amazon Kinesis Data Firehose delivery stream.
     *
     * @param records           a list of maps representing the records to be sent
     * @param batchSize         the maximum number of records to include in each
batch
     * @param deliveryStreamName the name of the Kinesis Data Firehose delivery
stream
     * @throws IllegalArgumentException if the input parameters are invalid (null or
empty)
     * @throws RuntimeException         if there is an error putting the record
batch
     */
    public static void putRecordBatch(List<Map<String, Object>> records, int
batchSize, String deliveryStreamName) {
        if (records == null || records.isEmpty() || deliveryStreamName == null ||
deliveryStreamName.isEmpty()) {
            throw new IllegalArgumentException("Invalid input: records or delivery
stream name cannot be null/empty");
        }
        ObjectMapper objectMapper = new ObjectMapper();

        try {
            for (int i = 0; i < records.size(); i += batchSize) {
                List<Map<String, Object>> batch = records.subList(i, Math.min(i +
batchSize, records.size()));

                List<Record> batchRecords = batch.stream().map(record -> {
                    try {
                        String jsonRecord = objectMapper.writeValueAsString(record);
                        return Record.builder()

.data(SdkBytes.fromByteArray(jsonRecord.getBytes(StandardCharsets.UTF_8)))
```

```
                            .build();
                    } catch (Exception e) {
                        throw new RuntimeException("Error creating Firehose record",
e);
                    }
                }).collect(Collectors.toList());

                PutRecordBatchRequest request = PutRecordBatchRequest.builder()
                    .deliveryStreamName(deliveryStreamName)
                    .records(batchRecords)
                    .build();

                PutRecordBatchResponse response =
getFirehoseClient().putRecordBatch(request);

                if (response.failedPutCount() > 0) {
                    response.requestResponses().stream()
                        .filter(r -> r.errorCode() != null)
                        .forEach(r -> System.err.println("Failed record: " +
r.errorMessage()));
                }
                System.out.println("Batch sent with size: " + batchRecords.size());
            }
        } catch (Exception e) {
            throw new RuntimeException("Failed to put record batch: " +
e.getMessage(), e);
        }
    }

    public static void monitorMetrics(String deliveryStreamName) {
        Instant endTime = Instant.now();
        Instant startTime = endTime.minusSeconds(600);

        List<String> metrics = List.of("IncomingBytes", "IncomingRecords",
"FailedPutCount");
        metrics.forEach(metric -> monitorMetric(metric, startTime, endTime,
deliveryStreamName));
    }

    private static void monitorMetric(String metricName, Instant startTime, Instant
endTime, String deliveryStreamName) {
        try {
            GetMetricStatisticsRequest request =
GetMetricStatisticsRequest.builder()
```

```
                .namespace("AWS/Firehose")
                .metricName(metricName)

 .dimensions(Dimension.builder().name("DeliveryStreamName").value(deliveryStreamName).build(
                .startTime(startTime)
                .endTime(endTime)
                .period(60)
                .statistics(Statistic.SUM)
                .build();

            GetMetricStatisticsResponse response =
 getCloudWatchClient().getMetricStatistics(request);
            double totalSum =
 response.datapoints().stream().mapToDouble(Datapoint::sum).sum();
            System.out.println(metricName + ": " + totalSum);
        } catch (Exception e) {
            System.err.println("Failed to monitor metric " + metricName + ": " +
 e.getMessage());
        }
    }

    public static String readJsonFile(String fileName) throws IOException {
        try (InputStream inputStream =
 FirehoseScenario.class.getResourceAsStream("/" + fileName);
             Scanner scanner = new Scanner(inputStream, StandardCharsets.UTF_8)) {
            return scanner.useDelimiter("\\\\A").next();
        } catch (Exception e) {
            throw new RuntimeException("Error reading file: " + fileName, e);
        }
    }

    private static void closeClients() {
        try {
            if (firehoseClient != null) firehoseClient.close();
            if (cloudWatchClient != null) cloudWatchClient.close();
        } catch (Exception e) {
            System.err.println("Error closing clients: " + e.getMessage());
        }
    }
}
```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella Documentazione di riferimento delle API AWS SDK for Java 2.x .
  - PutRecord
  - PutRecordBatch

Esempi di Amazon DocumentDB con SDK for Java 2.x

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando Amazon DocumentDB. AWS SDK for Java 2.x

Ogni esempio include un collegamento al codice sorgente completo, dove puoi trovare istruzioni su come configurare ed eseguire il codice nel contesto.

Argomenti

- Esempi serverless

Esempi serverless

Richiamare una funzione Lambda da un trigger Amazon DocumentDB

Il seguente esempio di codice mostra come implementare una funzione Lambda che riceve un evento attivato dalla ricezione di record da un flusso di modifiche di DocumentDB. La funzione recupera il payload DocumentDB 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 Amazon DocumentDB con Lambda tramite Java.

```
import java.util.List;
import java.util.Map;

import com.amazonaws.services.lambda.runtime.Context;
```

```
import com.amazonaws.services.lambda.runtime.RequestHandler;

public class Example implements RequestHandler<Map<String, Object>, String> {

    @SuppressWarnings("unchecked")
    @Override
    public String handleRequest(Map<String, Object> event, Context context) {
        List<Map<String, Object>> events = (List<Map<String, Object>>)
 event.get("events");
        for (Map<String, Object> record : events) {
            Map<String, Object> eventData = (Map<String, Object>)
 record.get("event");
            processEventData(eventData);
        }

        return "OK";
    }

    @SuppressWarnings("unchecked")
    private void processEventData(Map<String, Object> eventData) {
        String operationType = (String) eventData.get("operationType");
        System.out.println("operationType: %s".formatted(operationType));

        Map<String, Object> ns = (Map<String, Object>) eventData.get("ns");

        String db = (String) ns.get("db");
        System.out.println("db: %s".formatted(db));
        String coll = (String) ns.get("coll");
        System.out.println("coll: %s".formatted(coll));

        Map<String, Object> fullDocument = (Map<String, Object>)
 eventData.get("fullDocument");
        System.out.println("fullDocument: %s".formatted(fullDocument));
    }

}
```

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 nozioni di base sono esempi di codice che mostrano come eseguire le operazioni essenziali all'interno di un servizio.

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.

Gli scenari sono esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio o combinate con altri Servizi AWS.

AWS i contributi della community sono esempi che sono stati creati e gestiti da diversi team. AWS Per fornire feedback, utilizza il meccanismo fornito negli archivi collegati.

Ogni esempio include un collegamento al codice sorgente completo, dove puoi 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, consulta la ListTablessezione AWS SDK for Java 2.x API Reference.

Argomenti

- Nozioni di base
- Azioni
- Scenari
- Esempi serverless
- AWS contributi della comunità

Nozioni di base

Informazioni di base

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)
```

```
            .billingMode(BillingMode.PAY_PER_REQUEST) //  DynamoDB automatically
scales based on traffic.
            .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.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 * <p>
 * This Java example performs these tasks:
 * <p>
 * 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 {
        String tableName = "Movies";
        String fileName = "../../../resources/sample_files/movies.json";
        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)
            .billingMode(BillingMode.PAY_PER_REQUEST) //  DynamoDB automatically
scales based on traffic.
            .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

Azioni

`BatchGetItem`

Il seguente esempio di codice mostra come usare`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 BatchGetItemReference.

`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 BatchWriteItemin 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.BillingMode;
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.OnDemandThroughput;
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.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * 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())
            .billingMode(BillingMode.PAY_PER_REQUEST) //  DynamoDB automatically
scales based on traffic.
            .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, consulta la CreateTablesezione 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, consulta la DeleteItemsezione 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, consulta la DeleteTablesezione 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();

        describeDymamoDBTable(ddb, tableName);
        ddb.close();
    }

    public static void describeDymamoDBTable(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, consulta la DescribeTablesezione 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 utilizzando. AWS SDK for Java 2.x

```
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.logging.Level;
import java.util.logging.Logger;
```

```
   public DescribeTimeToLiveResponse describeTTL(final String tableName, final
Region region) {
       final DescribeTimeToLiveRequest request =
           DescribeTimeToLiveRequest.builder().tableName(tableName).build();

       try (DynamoDbClient ddb = dynamoDbClient != null
           ? dynamoDbClient
           : DynamoDbClient.builder().region(region).build()) {
           return ddb.describeTimeToLive(request);
       } catch (ResourceNotFoundException e) {
           System.err.format(TABLE_NOT_FOUND_ERROR, tableName);
           throw e;
       } catch (DynamoDbException e) {
           System.err.println(e.getMessage());
           throw e;
       }
   }
```

- Per i dettagli sull'API, consulta la sezione API DescribeTimeToLiveReference 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 GetItemin 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, consulta la ListTablessezione 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 PutItemin 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(expressionAttributesNames)
                    .expressionAttributeValues(expressionAttributeValues)
                    .build();

            System.out.println("=== Movie Titles ===");
            QueryResponse response = ddb.query(request);
            response.items()
                    .forEach(movie -> System.out.println(movie.get("title").s()));

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
```

```
    }
}
```

- 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, consulta la UpdateItemsezione 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 utilizzando. AWS SDK for Java 2.x

```
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.logging.Level;
import java.util.logging.Logger;

    public UpdateTimeToLiveResponse enableTTL(final String tableName, final String
 attributeName, final Region region) {
        final TimeToLiveSpecification ttlSpec = TimeToLiveSpecification.builder()
            .attributeName(attributeName)
            .enabled(true)
            .build();

        final UpdateTimeToLiveRequest request = UpdateTimeToLiveRequest.builder()
            .tableName(tableName)
            .timeToLiveSpecification(ttlSpec)
            .build();

        try (DynamoDbClient ddb = dynamoDbClient != null
            ? dynamoDbClient
            : DynamoDbClient.builder().region(region).build()) {
            return ddb.updateTimeToLive(request);
        } catch (ResourceNotFoundException e) {
            System.err.format(TABLE_NOT_FOUND_ERROR, tableName);
            throw e;
        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            throw e;
        }
    }
```

Disabilita il TTL su una tabella DynamoDB esistente utilizzando. AWS SDK for Java 2.x

```
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.logging.Level;
import java.util.logging.Logger;

    public UpdateTimeToLiveResponse disableTTL(
        final String tableName, final String attributeName, final Region region) {
        final TimeToLiveSpecification ttlSpec = TimeToLiveSpecification.builder()
            .attributeName(attributeName)
            .enabled(false)
            .build();

        final UpdateTimeToLiveRequest request = UpdateTimeToLiveRequest.builder()
            .tableName(tableName)
            .timeToLiveSpecification(ttlSpec)
            .build();

        try (DynamoDbClient ddb = dynamoDbClient != null
            ? dynamoDbClient
            : DynamoDbClient.builder().region(region).build()) {
            return ddb.updateTimeToLive(request);
        } catch (ResourceNotFoundException e) {
            System.err.format(TABLE_NOT_FOUND_ERROR, tableName);
            throw e;
        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            throw e;
        }
    }
```

- Per i dettagli sull'API, consulta la sezione API UpdateTimeToLiveReference AWS SDK for Java 2.x .

Scenari

Costruisci un'app per inviare dati a una tabella DynamoDB

Il seguente esempio di codice mostra come creare un'applicazione che invia dati a una tabella Amazon DynamoDB e ti avvisa quando un utente aggiorna la tabella.

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, consulta l'esempio completo su. GitHub

Servizi utilizzati in questo esempio

- DynamoDB
- Amazon SNS

Confronta più valori con un singolo attributo

Il seguente esempio di codice mostra come confrontare più valori con un singolo attributo in DynamoDB.

- Utilizzate l'operatore IN per confrontare più valori con un singolo attributo.
- Confronta l'operatore IN con più condizioni OR.
- Comprendi i vantaggi in termini di prestazioni e complessità di espressione derivanti dall'utilizzo di IN.

SDK per Java 2.x

Confronta più valori con un singolo attributo in DynamoDB utilizzando. AWS SDK for Java 2.x

```
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.ScanRequest;
import software.amazon.awssdk.services.dynamodb.model.ScanResponse;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Locale;
import java.util.Map;
```

```
   /**
    * Queries a table using the IN operator to compare multiple values with a
single attribute.
    *
    * <p>This method demonstrates how to use the IN operator in a filter expression
    * to match an attribute against multiple values.
    *
    * @param dynamoDbClient The DynamoDB client
    * @param tableName The name of the DynamoDB table
    * @param partitionKeyName The name of the partition key attribute
    * @param partitionKeyValue The value of the partition key to query
    * @param attributeName The name of the attribute to compare
    * @param valuesList List of values to compare against
    * @return The query response from DynamoDB
    * @throws DynamoDbException if an error occurs during the operation
    */
   public static QueryResponse compareMultipleValues(
       DynamoDbClient dynamoDbClient,
       String tableName,
       String partitionKeyName,
       AttributeValue partitionKeyValue,
       String attributeName,
       List<AttributeValue> valuesList) {

       // Create expression attribute names
       Map<String, String> expressionAttributeNames = new HashMap<>();
       expressionAttributeNames.put("#pkName", partitionKeyName);
       expressionAttributeNames.put("#attrName", attributeName);

       // Create expression attribute values
       Map<String, AttributeValue> expressionAttributeValues = new HashMap<>();
       expressionAttributeValues.put(":pkValue", partitionKeyValue);

       // Add values for IN operator
       for (int i = 0; i < valuesList.size(); i++) {
           expressionAttributeValues.put(":val" + i, valuesList.get(i));
       }

       // Build the IN clause
       StringBuilder inClause = new StringBuilder();
       for (int i = 0; i < valuesList.size(); i++) {
           if (i > 0) {
               inClause.append(", ");
```

```
            }
            inClause.append(":val").append(i);
        }

        // Define the query parameters
        QueryRequest request = QueryRequest.builder()
            .tableName(tableName)
            .keyConditionExpression("#pkName = :pkValue")
            .filterExpression("#attrName IN (" + inClause.toString() + ")")
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .build();

        // Perform the query operation
        return dynamoDbClient.query(request);
    }

    /**
     * Queries a table using multiple OR conditions to compare multiple values with
a single attribute.
     *
     * <p>This method demonstrates the alternative approach to using the IN
operator,
     * by using multiple OR conditions.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param partitionKeyName The name of the partition key attribute
     * @param partitionKeyValue The value of the partition key to query
     * @param attributeName The name of the attribute to compare
     * @param valuesList List of values to compare against
     * @return The query response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static QueryResponse compareWithOrConditions(
        DynamoDbClient dynamoDbClient,
        String tableName,
        String partitionKeyName,
        AttributeValue partitionKeyValue,
        String attributeName,
        List<AttributeValue> valuesList) {

        // Create expression attribute names
        Map<String, String> expressionAttributeNames = new HashMap<>();
```

```
        expressionAttributeNames.put("#pkName", partitionKeyName);
        expressionAttributeNames.put("#attrName", attributeName);

        // Create expression attribute values
        Map<String, AttributeValue> expressionAttributeValues = new HashMap<>();
        expressionAttributeValues.put(":pkValue", partitionKeyValue);

        // Add values for OR conditions
        for (int i = 0; i < valuesList.size(); i++) {
            expressionAttributeValues.put(":val" + i, valuesList.get(i));
        }

        // Build the OR conditions
        StringBuilder orConditions = new StringBuilder();
        for (int i = 0; i < valuesList.size(); i++) {
            if (i > 0) {
                orConditions.append(" OR ");
            }
            orConditions.append("#attrName = :val").append(i);
        }

        // Define the query parameters
        QueryRequest request = QueryRequest.builder()
            .tableName(tableName)
            .keyConditionExpression("#pkName = :pkValue")
            .filterExpression(orConditions.toString())
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .build();

        // Perform the query operation
        return dynamoDbClient.query(request);
    }

    /**
     * Compares the performance of using the IN operator versus multiple OR
conditions.
     *
     * <p>This method demonstrates the performance difference between using the IN
operator
     * and using multiple OR conditions.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
```

```
     * @param partitionKeyName The name of the partition key attribute
     * @param partitionKeyValue The value of the partition key to query
     * @param attributeName The name of the attribute to compare
     * @param valuesList List of values to compare against
     * @return Map containing the performance comparison results
     */
    public static Map<String, Object> comparePerformance(
        DynamoDbClient dynamoDbClient,
        String tableName,
        String partitionKeyName,
        AttributeValue partitionKeyValue,
        String attributeName,
        List<AttributeValue> valuesList) {

        Map<String, Object> results = new HashMap<>();

        try {
            // Measure performance of IN operator
            long inStartTime = System.nanoTime();
            QueryResponse inResponse = compareMultipleValues(
                dynamoDbClient, tableName, partitionKeyName, partitionKeyValue,
attributeName, valuesList);
            long inEndTime = System.nanoTime();
            long inDuration = inEndTime - inStartTime;

            // Measure performance of OR conditions
            long orStartTime = System.nanoTime();
            QueryResponse orResponse = compareWithOrConditions(
                dynamoDbClient, tableName, partitionKeyName, partitionKeyValue,
attributeName, valuesList);
            long orEndTime = System.nanoTime();
            long orDuration = orEndTime - orStartTime;

            // Record results
            results.put("inOperatorDuration", inDuration);
            results.put("orConditionsDuration", orDuration);
            results.put("inOperatorItems", inResponse.count());
            results.put("orConditionsItems", orResponse.count());
            results.put("inOperatorExpression", "IN operator with " +
valuesList.size() + " values");
            results.put("orConditionsExpression", valuesList.size() + " OR
conditions");
            results.put("success", true);
```

```
        } catch (DynamoDbException e) {
            results.put("success", false);
            results.put("error", e.getMessage());
        }

        return results;
    }

    /**
     * Scans a table using the IN operator with a large number of values.
     *
     * <p>This method demonstrates how to use the IN operator with a large number of
values,
     * which can help stay within the 300 operator limit.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param attributeName The name of the attribute to compare
     * @param valuesList List of values to compare against
     * @return The scan response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static ScanResponse scanWithLargeInClause(
        DynamoDbClient dynamoDbClient, String tableName, String attributeName,
List<AttributeValue> valuesList) {

        // Create expression attribute names
        Map<String, String> expressionAttributeNames = new HashMap<>();
        expressionAttributeNames.put("#attrName", attributeName);

        // Create expression attribute values
        Map<String, AttributeValue> expressionAttributeValues = new HashMap<>();

        // Add values for IN operator
        for (int i = 0; i < valuesList.size(); i++) {
            expressionAttributeValues.put(":val" + i, valuesList.get(i));
        }

        // Build the IN clause
        StringBuilder inClause = new StringBuilder();
        for (int i = 0; i < valuesList.size(); i++) {
            if (i > 0) {
                inClause.append(", ");
            }
```

```
            inClause.append(":val").append(i);
        }

        // Define the scan parameters
        ScanRequest request = ScanRequest.builder()
            .tableName(tableName)
            .filterExpression("#attrName IN (" + inClause.toString() + ")")
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .build();

        // Perform the scan operation
        return dynamoDbClient.scan(request);
    }

    /**
     * Generates a list of sample values for testing.
     *
     * <p>Helper method to generate a list of sample values for testing.
     *
     * @param valueType The type of values to generate (string, number, or boolean)
     * @param count The number of values to generate
     * @return List of generated attribute values
     */
    public static List<AttributeValue> generateSampleValues(String valueType, int
count) {
        List<AttributeValue> values = new ArrayList<>();

        for (int i = 0; i < count; i++) {
            AttributeValue value;

            switch (valueType.toLowerCase(Locale.ROOT)) {
                case "string":
                    value = AttributeValue.builder().s("Value" + i).build();
                    break;
                case "number":
                    value = AttributeValue.builder().n(String.valueOf(i)).build();
                    break;
                case "boolean":
                    value = AttributeValue.builder().bool(i % 2 == 0).build();
                    break;
                default:
                    throw new IllegalArgumentException("Unsupported value type: " +
valueType);
```

```
            }

            values.add(value);
        }

        return values;
    }
```

Esempio di utilizzo del confronto di più valori con. AWS SDK for Java 2.x

```
    public static void exampleUsage(DynamoDbClient dynamoDbClient, String tableName)
{
        System.out.println("Demonstrating how to compare multiple values with a
single attribute in DynamoDB");

        try {
            // Example 1: Using the IN operator
            System.out.println("\nExample 1: Using the IN operator");
            List<AttributeValue> categories = List.of(
                AttributeValue.builder().s("Electronics").build(),
                AttributeValue.builder().s("Computers").build(),
                AttributeValue.builder().s("Accessories").build());

            QueryResponse inResponse = compareMultipleValues(
                dynamoDbClient,
                tableName,
                "Department",
                AttributeValue.builder().s("Retail").build(),
                "Category",
                categories);

            System.out.println("Found " + inResponse.count() + " items using IN
operator");
            System.out.println("Items: " + inResponse.items());

            // Example 2: Using multiple OR conditions
            System.out.println("\nExample 2: Using multiple OR conditions");
            QueryResponse orResponse = compareWithOrConditions(
                dynamoDbClient,
                tableName,
                "Department",
                AttributeValue.builder().s("Retail").build(),
```

```
                "Category",
                categories);

            System.out.println("Found " + orResponse.count() + " items using OR
conditions");
            System.out.println("Items: " + orResponse.items());

            // Example 3: Performance comparison
            System.out.println("\nExample 3: Performance comparison");
            Map<String, Object> perfComparison = comparePerformance(
                dynamoDbClient,
                tableName,
                "Department",
                AttributeValue.builder().s("Retail").build(),
                "Category",
                categories);

            if ((boolean) perfComparison.get("success")) {
                System.out.println("IN operator duration: " +
perfComparison.get("inOperatorDuration") + " ns");
                System.out.println("OR conditions duration: " +
perfComparison.get("orConditionsDuration") + " ns");
                System.out.println("IN operator found " +
perfComparison.get("inOperatorItems") + " items");
                System.out.println("OR conditions found " +
perfComparison.get("orConditionsItems") + " items");
                System.out.println("Expression complexity comparison:");
                System.out.println("  IN operator: " +
perfComparison.get("inOperatorExpression"));
                System.out.println("  OR conditions: " +
perfComparison.get("orConditionsExpression"));
            } else {
                System.out.println("Performance comparison failed: " +
perfComparison.get("error"));
            }

            // Example 4: Using IN with a large number of values
            System.out.println("\nExample 4: Using IN with a large number of
values");
            List<AttributeValue> productIds = generateSampleValues("string", 20);

            ScanResponse largeInResponse = scanWithLargeInClause(dynamoDbClient,
tableName, "ProductId", productIds);
```

```
            System.out.println(
                "Found " + largeInResponse.count() + " items using IN with " +
productIds.size() + " values");

            // Explain the benefits of using IN
            System.out.println("\nKey points about using the IN operator in
DynamoDB:");
            System.out.println("1. The IN operator allows comparing a single
attribute against multiple values");
            System.out.println("2. IN is more concise than using multiple OR
conditions");
            System.out.println("3. IN counts as only 1 operator regardless of the
number of values");
            System.out.println("4. Multiple OR conditions count as 1 operator per
condition plus 1 per OR");
            System.out.println("5. Using IN helps stay within the 300 operator limit
for complex expressions");
            System.out.println("6. IN can be used in filter expressions and
condition expressions");
            System.out.println("7. The IN operator supports up to 100 comparison
values");

        } catch (DynamoDbException e) {
            System.err.println("Error: " + 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 .
  - Query
  - Scan

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

Aggiorna il TTL su un elemento DynamoDB esistente in una tabella, con una condizione.

```
package com.amazon.samplelib.ttl;

import com.amazon.samplelib.CodeSampleUtils;
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.ConditionalCheckFailedException;
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 java.util.Map;
import java.util.Optional;

/**
 * Updates an item in a DynamoDB table with TTL attributes using a conditional
 expression.
 * This class demonstrates how to conditionally update TTL expiration timestamps.
 */
public class UpdateTTLConditional {

    private static 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)
            """;
    private static final int DAYS_TO_EXPIRE = 90;
    private static final int SECONDS_PER_DAY = 24 * 60 * 60;
    private static final String PRIMARY_KEY_ATTR = "primaryKey";
    private static final String SORT_KEY_ATTR = "sortKey";
    private static final String UPDATED_AT_ATTR = "updatedAt";
    private static final String EXPIRE_AT_ATTR = "expireAt";
```

```
   private static final String UPDATE_EXPRESSION = "SET " + UPDATED_AT_ATTR + "=:c,
" + EXPIRE_AT_ATTR + "=:e";
   private static final String CONDITION_EXPRESSION = "attribute_exists(" +
PRIMARY_KEY_ATTR + ")";
   private static final String SUCCESS_MESSAGE = "%s UpdateItem operation with TTL
successful.";
   private static final String CONDITION_FAILED_MESSAGE = "Condition check failed.
Item does not exist.";
   private static final String TABLE_NOT_FOUND_ERROR = "Error: The Amazon DynamoDB
table \"%s\" can't be found.";

   private final DynamoDbClient dynamoDbClient;

   /**
    * Constructs an UpdateTTLConditional with a default DynamoDB client.
    */
   public UpdateTTLConditional() {
       this.dynamoDbClient = null;
   }

   /**
    * Constructs an UpdateTTLConditional with the specified DynamoDB client.
    *
    * @param dynamoDbClient The DynamoDB client to use
    */
   public UpdateTTLConditional(final DynamoDbClient dynamoDbClient) {
       this.dynamoDbClient = dynamoDbClient;
   }

   /**
    * Main method to demonstrate conditionally updating an item with TTL.
    *
    * @param args Command line arguments
    */
   public static void main(final String[] args) {
       try {
           int result = new UpdateTTLConditional().processArgs(args);
           System.exit(result);
       } catch (Exception e) {
           System.err.println(e.getMessage());
           System.exit(1);
       }
   }
```

```
    /**
     * Process command line arguments and conditionally update an item with TTL.
     *
     * @param args Command line arguments
     * @return 0 if successful, non-zero otherwise
     * @throws ResourceNotFoundException If the table doesn't exist
     * @throws DynamoDbException If an error occurs during the operation
     * @throws IllegalArgumentException If arguments are invalid
     */
    public int processArgs(final String[] args) {
        // Argument validation (remove or replace this line when reusing this code)
        CodeSampleUtils.validateArgs(args, new int[] {3, 4}, USAGE);

        final String tableName = args[0];
        final String primaryKey = args[1];
        final String sortKey = args[2];
        final Region region = Optional.ofNullable(args.length > 3 ? args[3] : null)
            .map(Region::of)
            .orElse(Region.US_EAST_1);

        // 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 + (DAYS_TO_EXPIRE * SECONDS_PER_DAY);

        // Create the key map for the item to update
        final Map<String, AttributeValue> keyMap = Map.of(
            PRIMARY_KEY_ATTR, AttributeValue.builder().s(primaryKey).build(),
            SORT_KEY_ATTR, AttributeValue.builder().s(sortKey).build());

        // Create the expression attribute values
        final Map<String, AttributeValue> expressionAttributeValues = Map.of(
            ":c", AttributeValue.builder().n(String.valueOf(currentTime)).build(),
            ":e", AttributeValue.builder().n(String.valueOf(expireDate)).build());

        final UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(keyMap)
            .updateExpression(UPDATE_EXPRESSION)
            .conditionExpression(CONDITION_EXPRESSION)
            .expressionAttributeValues(expressionAttributeValues)
            .build();
```

```
        try (DynamoDbClient ddb = dynamoDbClient != null
            ? dynamoDbClient
            : DynamoDbClient.builder().region(region).build()) {
            final UpdateItemResponse response = ddb.updateItem(request);
            System.out.println(String.format(SUCCESS_MESSAGE, tableName));
            return 0;
        } catch (ConditionalCheckFailedException e) {
            System.err.println(CONDITION_FAILED_MESSAGE);
            throw e;
        } catch (ResourceNotFoundException e) {
            System.err.format(TABLE_NOT_FOUND_ERROR, tableName);
            throw e;
        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            throw e;
        }
    }
}
```

- Per i dettagli sull'API, consulta la sezione API UpdateItemReference AWS SDK for Java 2.x .

Conta gli operatori di espressione

Il seguente esempio di codice mostra come contare gli operatori di espressione in DynamoDB.

- Comprendi il limite di 300 operatori di DynamoDB.
- Conta gli operatori nelle espressioni complesse.
- Ottimizza le espressioni per rimanere entro i limiti.

SDK per Java 2.x

Dimostra il conteggio degli operatori di espressioni utilizzando AWS SDK for Java 2.x.

```
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.UpdateItemRequest;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemResponse;

import java.util.HashMap;
```

```
import java.util.Map;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

    /**
     * Creates a complex filter expression with a specified number of conditions.
     *
     * <p>This method demonstrates how to generate a complex expression with
     * a specific number of operators to test the 300 operator limit.
     *
     * @param conditionsCount Number of conditions to include
     * @param useAnd Whether to use AND (true) or OR (false) between conditions
     * @return Map containing the filter expression, attribute values, and operator
 count
     */
    public static Map<String, Object> createComplexFilterExpression(int
 conditionsCount, boolean useAnd) {
        // Initialize the expression parts and attribute values
        StringBuilder filterExpression = new StringBuilder();
        Map<String, AttributeValue> expressionAttributeValues = new HashMap<>();

        // Generate the specified number of conditions
        for (int i = 0; i < conditionsCount; i++) {
            // Add the operator between conditions (except for the first one)
            if (i > 0) {
                filterExpression.append(useAnd ? " AND " : " OR ");
            }

            // Alternate between different comparison operators for variety
            String valueKey = ":val" + i;

            switch (i % 5) {
                case 0:
                    filterExpression.append("attribute").append(i).append(" =
 ").append(valueKey);
                    expressionAttributeValues.put(
                        valueKey, AttributeValue.builder().s("value" + i).build());
                    break;
                case 1:
                    filterExpression.append("attribute").append(i).append(" >
 ").append(valueKey);
                    expressionAttributeValues.put(
                        valueKey,
 AttributeValue.builder().n(String.valueOf(i)).build());
```

```
                    break;
                case 2:
                    filterExpression.append("attribute").append(i).append(" <
").append(valueKey);
                    expressionAttributeValues.put(
                        valueKey,
                        AttributeValue.builder().n(String.valueOf(i * 10)).build());
                    break;
                case 3:
                    filterExpression
                        .append("contains(attribute")
                        .append(i)
                        .append(", ")
                        .append(valueKey)
                        .append(")");
                    expressionAttributeValues.put(
                        valueKey, AttributeValue.builder().s("substring" +
i).build());
                    break;
                case 4:
                    filterExpression
                        .append("attribute_exists(attribute")
                        .append(i)
                        .append(")");
                    break;
                default:
                    // This case will never be reached, but added to satisfy
checkstyle
                    break;
            }
        }

        // Calculate the operator count
        // Each condition has 1 operator (=, >, <, contains, attribute_exists)
        // Each AND or OR between conditions is 1 operator
        int operatorCount = conditionsCount + (conditionsCount > 0 ? conditionsCount
- 1 : 0);

        // Create the result map
        Map<String, Object> result = new HashMap<>();
        result.put("filterExpression", filterExpression.toString());
        result.put("expressionAttributeValues", expressionAttributeValues);
        result.put("operatorCount", operatorCount);
```

```
        return result;
    }

    /**
     * Creates a complex update expression with a specified number of operations.
     *
     * <p>This method demonstrates how to generate a complex update expression with
     * a specific number of operators to test the 300 operator limit.
     *
     * @param operationsCount Number of operations to include
     * @return Map containing the update expression, attribute values, and operator
count
     */
    public static Map<String, Object> createComplexUpdateExpression(int
operationsCount) {
        // Initialize the expression parts and attribute values
        StringBuilder updateExpression = new StringBuilder("SET ");
        Map<String, AttributeValue> expressionAttributeValues = new HashMap<>();

        // Generate the specified number of SET operations
        for (int i = 0; i < operationsCount; i++) {
            // Add comma between operations (except for the first one)
            if (i > 0) {
                updateExpression.append(", ");
            }

            // Alternate between different types of SET operations
            String valueKey = ":val" + i;

            switch (i % 3) {
                case 0:
                    // Simple assignment (1 operator: =)
                    updateExpression.append("attribute").append(i).append(" =
").append(valueKey);
                    expressionAttributeValues.put(
                        valueKey, AttributeValue.builder().s("value" + i).build());
                    break;
                case 1:
                    // Addition (2 operators: = and +)
                    updateExpression
                        .append("attribute")
                        .append(i)
                        .append(" = attribute")
                        .append(i)
```

```
                        .append(" + ")
                        .append(valueKey);
                    expressionAttributeValues.put(
                        valueKey,
AttributeValue.builder().n(String.valueOf(i)).build());
                    break;
                case 2:
                    // Conditional assignment with if_not_exists (2 operators: = and
if_not_exists)
                    updateExpression
                        .append("attribute")
                        .append(i)
                        .append(" = if_not_exists(attribute")
                        .append(i)
                        .append(", ")
                        .append(valueKey)
                        .append(")");
                    expressionAttributeValues.put(
                        valueKey,
                        AttributeValue.builder().n(String.valueOf(i * 10)).build());
                    break;
                default:
                    // This case will never be reached, but added to satisfy
checkstyle
                    break;
            }
        }

        // Calculate the operator count
        // Each operation has 1-2 operators as noted above
        int operatorCount = 0;
        for (int i = 0; i < operationsCount; i++) {
            operatorCount += (i % 3 == 0) ? 1 : 2;
        }

        // Create the result map
        Map<String, Object> result = new HashMap<>();
        result.put("updateExpression", updateExpression.toString());
        result.put("expressionAttributeValues", expressionAttributeValues);
        result.put("operatorCount", operatorCount);

        return result;
    }
```

```
    /**
     * Test the operator limit by attempting an operation with a complex expression.
     *
     * <p>This method demonstrates what happens when an expression approaches or
     * exceeds the 300 operator limit.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param operatorCount Target number of operators to include
     * @return Map containing the result of the operation attempt
     */
    public static Map<String, Object> testOperatorLimit(
        DynamoDbClient dynamoDbClient, String tableName, Map<String, AttributeValue>
key, int operatorCount) {

        // Create a complex update expression with the specified operator count
        Map<String, Object> expressionData =
            createComplexUpdateExpression((int) Math.ceil(operatorCount / 1.5)); //
Adjust to get close to target count

        String updateExpression = (String) expressionData.get("updateExpression");
        @SuppressWarnings("unchecked")
        Map<String, AttributeValue> expressionAttributeValues =
            (Map<String, AttributeValue>)
expressionData.get("expressionAttributeValues");
        int actualCount = (int) expressionData.get("operatorCount");

        System.out.println("Generated update expression with approximately " +
actualCount + " operators");

        // Define the update parameters
        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .updateExpression(updateExpression)
            .expressionAttributeValues(expressionAttributeValues)
            .returnValues("UPDATED_NEW")
            .build();

        try {
            // Attempt the update operation
            UpdateItemResponse response = dynamoDbClient.updateItem(request);
```

```
            Map<String, Object> result = new HashMap<>();
            result.put("success", true);
            result.put("message", "Operation succeeded with " + actualCount + "
operators");
            result.put("data", response);
            return result;

        } catch (DynamoDbException e) {
            // Check if the error is due to exceeding the operator limit
            if (e.getMessage().contains("too many operators")) {
                Map<String, Object> result = new HashMap<>();
                result.put("success", false);
                result.put("message", "Operation failed: " + e.getMessage());
                result.put("operatorCount", actualCount);
                return result;
            }

            // Return other errors
            Map<String, Object> result = new HashMap<>();
            result.put("success", false);
            result.put("message", "Operation failed: " + e.getMessage());
            result.put("error", e);
            return result;
        }
    }

    /**
     * Break down a complex expression into multiple simpler operations.
     *
     * <p>This method demonstrates how to handle expressions that would exceed
     * the 300 operator limit by breaking them into multiple operations.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param totalOperations Total number of operations to perform
     * @return Map containing the results of the operations
     */
    public static Map<String, Object> breakDownComplexExpression(
        DynamoDbClient dynamoDbClient, String tableName, Map<String, AttributeValue>
key, int totalOperations) {

        // Calculate how many operations we can safely include in each batch
        // Using 150 as a conservative limit (well below 300)
```

```
        final int operationsPerBatch = 100;
        final int batchCount = (int) Math.ceil((double) totalOperations /
operationsPerBatch);

        System.out.println("Breaking down " + totalOperations + " operations into "
+ batchCount + " batches");

        Map<String, Object> results = new HashMap<>();
        results.put("totalBatches", batchCount);

        Map<Integer, Map<String, Object>> batchResults = new HashMap<>();

        // Process each batch
        for (int batch = 0; batch < batchCount; batch++) {
            // Calculate the operations for this batch
            int batchStart = batch * operationsPerBatch;
            int batchEnd = Math.min(batchStart + operationsPerBatch,
totalOperations);
            int batchSize = batchEnd - batchStart;

            System.out.println(
                "Processing batch " + (batch + 1) + "/" + batchCount + " with " +
batchSize + " operations");

            // Create an update expression for this batch
            Map<String, Object> expressionData =
createComplexUpdateExpression(batchSize);

            String updateExpression = (String)
expressionData.get("updateExpression");
            @SuppressWarnings("unchecked")
            Map<String, AttributeValue> expressionAttributeValues =
                (Map<String, AttributeValue>)
expressionData.get("expressionAttributeValues");
            int operatorCount = (int) expressionData.get("operatorCount");

            // Define the update parameters
            UpdateItemRequest request = UpdateItemRequest.builder()
                .tableName(tableName)
                .key(key)
                .updateExpression(updateExpression)
                .expressionAttributeValues(expressionAttributeValues)
                .returnValues("UPDATED_NEW")
                .build();
```

```
            try {
                // Perform the update operation for this batch
                UpdateItemResponse response = dynamoDbClient.updateItem(request);

                Map<String, Object> batchResult = new HashMap<>();
                batchResult.put("batch", batch + 1);
                batchResult.put("success", true);
                batchResult.put("operatorCount", operatorCount);
                batchResult.put("attributes", response.attributes());

                batchResults.put(batch, batchResult);

            } catch (DynamoDbException e) {
                Map<String, Object> batchResult = new HashMap<>();
                batchResult.put("batch", batch + 1);
                batchResult.put("success", false);
                batchResult.put("operatorCount", operatorCount);
                batchResult.put("error", e.getMessage());

                batchResults.put(batch, batchResult);

                // Continue with next batch instead of breaking
                continue;
            }
        }

        results.put("results", batchResults);
        return results;
    }

    /**
     * Count operators in a DynamoDB expression based on the rules in the
documentation.
     *
     * <p>This method demonstrates how operators are counted according to the
     * DynamoDB documentation.
     *
     * @param expression The DynamoDB expression to analyze
     * @return Map containing the breakdown of operator counts
     */
    public static Map<String, Integer> countOperatorsInExpression(String expression)
{
        // Initialize counters for different operator types
```

```
        Map<String, Integer> counts = new HashMap<>();
        counts.put("comparisonOperators", 0);
        counts.put("logicalOperators", 0);
        counts.put("functions", 0);
        counts.put("arithmeticOperators", 0);
        counts.put("specialOperators", 0);
        counts.put("total", 0);

        // Count comparison operators (=, <>, <, <=, >, >=)
        // This is a simplified approach and may not catch all cases
        int comparisonCount = 0;
        Pattern comparisonPattern = Pattern.compile("(=|<>|<=|>=|<|>)");
        Matcher comparisonMatcher = comparisonPattern.matcher(expression);
        while (comparisonMatcher.find()) {
            comparisonCount++;
        }
        counts.put("comparisonOperators", comparisonCount);

        // Count logical operators (AND, OR, NOT)
        int andCount = countOccurrences(expression, "\\bAND\\b");
        int orCount = countOccurrences(expression, "\\bOR\\b");
        int notCount = countOccurrences(expression, "\\bNOT\\b");
        counts.put("logicalOperators", andCount + orCount + notCount);

        // Count functions (attribute_exists, attribute_not_exists, attribute_type,
 begins_with, contains, size)
        int functionCount = countOccurrences(
            expression,
            "\\b(attribute_exists|attribute_not_exists|attribute_type|begins_with|
contains|size|if_not_exists)\\(");
        counts.put("functions", functionCount);

        // Count arithmetic operators (+ and -)
        // This is a simplified approach and may not catch all cases
        int arithmeticCount = 0;
        Pattern arithmeticPattern = Pattern.compile("[a-zA-Z0-9_)\\]]\\s*[\\+\\-]\
\s*[a-zA-Z0-9_:(]");
        Matcher arithmeticMatcher = arithmeticPattern.matcher(expression);
        while (arithmeticMatcher.find()) {
            arithmeticCount++;
        }
        counts.put("arithmeticOperators", arithmeticCount);

        // Count special operators (BETWEEN, IN)
```

```
        int betweenCount = countOccurrences(expression, "\\bBETWEEN\\b");
        int inCount = countOccurrences(expression, "\\bIN\\b");
        counts.put("specialOperators", betweenCount + inCount);

        // Add extra operators for BETWEEN (each BETWEEN includes an AND)
        int currentLogicalOps = counts.getOrDefault("logicalOperators", 0);
        counts.put("logicalOperators", currentLogicalOps + betweenCount);

        // Calculate total
        int total = counts.getOrDefault("comparisonOperators", 0)
            + counts.getOrDefault("logicalOperators", 0)
            + counts.getOrDefault("functions", 0)
            + counts.getOrDefault("arithmeticOperators", 0)
            + counts.getOrDefault("specialOperators", 0);
        counts.put("total", total);

        return counts;
    }

    /**
     * Helper method to count occurrences of a pattern in a string.
     *
     * @param text The text to search in
     * @param regex The regular expression pattern to search for
     * @return The number of occurrences
     */
    private static int countOccurrences(String text, String regex) {
        final Pattern pattern = Pattern.compile(regex);
        final Matcher matcher = pattern.matcher(text);
        int count = 0;
        while (matcher.find()) {
            count++;
        }
        return count;
    }
```

Esempio di utilizzo dell'operatore di espressione che conta con. AWS SDK for Java 2.x

```
    public static void exampleUsage(DynamoDbClient dynamoDbClient, String tableName)
{
        // Example key
        Map<String, AttributeValue> key = new HashMap<>();
```

```
        key.put("ProductId", AttributeValue.builder().s("P12345").build());

        System.out.println("Demonstrating DynamoDB expression operator counting and
the 300 operator limit");

        try {
            // Example 1: Analyze a simple expression
            System.out.println("\nExample 1: Analyzing a simple expression");
            String simpleExpression = "Price = :price AND Rating > :rating AND
Category IN (:cat1, :cat2, :cat3)";
            Map<String, Integer> simpleCount =
countOperatorsInExpression(simpleExpression);

            System.out.println("Expression: " + simpleExpression);
            System.out.println("Operator count breakdown:");
            System.out.println("- Comparison operators: " +
simpleCount.get("comparisonOperators"));
            System.out.println("- Logical operators: " +
simpleCount.get("logicalOperators"));
            System.out.println("- Functions: " + simpleCount.get("functions"));
            System.out.println("- Arithmetic operators: " +
simpleCount.get("arithmeticOperators"));
            System.out.println("- Special operators: " +
simpleCount.get("specialOperators"));
            System.out.println("- Total operators: " + simpleCount.get("total"));

            // Example 2: Analyze a complex expression
            System.out.println("\nExample 2: Analyzing a complex expression");
            String complexExpression = "(attribute_exists(Category) AND Size
BETWEEN :min AND :max) OR "
                + "(Price > :price AND contains(Description, :keyword) AND "
                + "(Rating >= :minRating OR Reviews > :minReviews))";
            Map<String, Integer> complexCount =
countOperatorsInExpression(complexExpression);

            System.out.println("Expression: " + complexExpression);
            System.out.println("Operator count breakdown:");
            System.out.println("- Comparison operators: " +
complexCount.get("comparisonOperators"));
            System.out.println("- Logical operators: " +
complexCount.get("logicalOperators"));
            System.out.println("- Functions: " + complexCount.get("functions"));
            System.out.println("- Arithmetic operators: " +
complexCount.get("arithmeticOperators"));
```

```
            System.out.println("- Special operators: " +
complexCount.get("specialOperators"));
            System.out.println("- Total operators: " + complexCount.get("total"));

            // Example 3: Test approaching the operator limit
            System.out.println("\nExample 3: Testing an expression approaching the
operator limit");
            Map<String, Object> approachingLimit = testOperatorLimit(dynamoDbClient,
tableName, key, 290);
            System.out.println(approachingLimit.get("message"));

            // Example 4: Test exceeding the operator limit
            System.out.println("\nExample 4: Testing an expression exceeding the
operator limit");
            Map<String, Object> exceedingLimit = testOperatorLimit(dynamoDbClient,
tableName, key, 310);
            System.out.println(exceedingLimit.get("message"));

            // Example 5: Breaking down a complex expression
            System.out.println("\nExample 5: Breaking down a complex expression into
multiple operations");
            Map<String, Object> breakdownResult =
breakDownComplexExpression(dynamoDbClient, tableName, key, 500);
            @SuppressWarnings("unchecked")
            Map<Integer, Map<String, Object>> results =
                (Map<Integer, Map<String, Object>>) breakdownResult.get("results");
            System.out.println(
                "Processed " + results.size() + " of " +
breakdownResult.get("totalBatches") + " batches");

            // Explain the operator counting rules
            System.out.println("\nKey points about DynamoDB expression operator
counting:");
            System.out.println("1. The maximum number of operators in any expression
is 300");
            System.out.println("2. Each comparison operator (=, <>, <, <=, >, >=)
counts as 1 operator");
            System.out.println("3. Each logical operator (AND, OR, NOT) counts as 1
operator");
            System.out.println("4. Each function call (attribute_exists, contains,
etc.) counts as 1 operator");
            System.out.println("5. Each arithmetic operator (+ or -) counts as 1
operator");
```

```
            System.out.println("6. BETWEEN counts as 2 operators (BETWEEN itself and
the AND within it)");
            System.out.println("7. IN counts as 1 operator regardless of the number
of values");
            System.out.println("8. Parentheses for grouping and attribute paths
don't count as operators");
            System.out.println("9. When you exceed the limit, the error always
reports '301 operators'");
            System.out.println("10. For complex operations, break them into multiple
smaller operations");

        } catch (Exception e) {
            System.err.println("Error: " + e.getMessage());
            e.printStackTrace();
        }
    }
```

- Per i dettagli sull'API, consulta la sezione UpdateItem AWS SDK for Java 2.xAPI Reference.

Creazione di un'applicazione serverless per gestire foto

Nell'esempio di codice seguente viene illustrato come creare un'applicazione serverless che consente agli utenti di gestire le foto mediante 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

Crea una tabella con un indice secondario globale

Il seguente esempio di codice mostra come creare una tabella con un indice secondario globale.

SDK per Java 2.x

Crea una tabella DynamoDB con Global Secondary Index utilizzando. AWS SDK for Java 2.x

```
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
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.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.CreateTableRequest;
import software.amazon.awssdk.services.dynamodb.model.DeleteTableRequest;
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.GlobalSecondaryIndex;
import software.amazon.awssdk.services.dynamodb.model.KeySchemaElement;
import software.amazon.awssdk.services.dynamodb.model.KeyType;
import software.amazon.awssdk.services.dynamodb.model.Projection;
import software.amazon.awssdk.services.dynamodb.model.ProjectionType;
import software.amazon.awssdk.services.dynamodb.model.ProvisionedThroughput;
import software.amazon.awssdk.services.dynamodb.model.PutItemRequest;
import software.amazon.awssdk.services.dynamodb.model.QueryRequest;
import software.amazon.awssdk.services.dynamodb.model.QueryResponse;
import software.amazon.awssdk.services.dynamodb.model.ScalarAttributeType;
import software.amazon.awssdk.services.dynamodb.waiters.DynamoDbWaiter;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

    public void createTable() {
        try {
            // Attribute definitions
            final List<AttributeDefinition> attributeDefinitions = new
 ArrayList<>();
```

```
        attributeDefinitions.add(AttributeDefinition.builder()
            .attributeName(ISSUE_ID_ATTR)
            .attributeType(ScalarAttributeType.S)
            .build());
        attributeDefinitions.add(AttributeDefinition.builder()
            .attributeName(TITLE_ATTR)
            .attributeType(ScalarAttributeType.S)
            .build());
        attributeDefinitions.add(AttributeDefinition.builder()
            .attributeName(CREATE_DATE_ATTR)
            .attributeType(ScalarAttributeType.S)
            .build());
        attributeDefinitions.add(AttributeDefinition.builder()
            .attributeName(DUE_DATE_ATTR)
            .attributeType(ScalarAttributeType.S)
            .build());

        // Key schema for table
        final List<KeySchemaElement> tableKeySchema = new ArrayList<>();
        tableKeySchema.add(KeySchemaElement.builder()
            .attributeName(ISSUE_ID_ATTR)
            .keyType(KeyType.HASH)
            .build()); // Partition key
        tableKeySchema.add(KeySchemaElement.builder()
            .attributeName(TITLE_ATTR)
            .keyType(KeyType.RANGE)
            .build()); // Sort key

        // Initial provisioned throughput settings for the indexes
        final ProvisionedThroughput ptIndex = ProvisionedThroughput.builder()
            .readCapacityUnits(1L)
            .writeCapacityUnits(1L)
            .build();

        // CreateDateIndex
        final List<KeySchemaElement> createDateKeySchema = new ArrayList<>();
        createDateKeySchema.add(KeySchemaElement.builder()
            .attributeName(CREATE_DATE_ATTR)
            .keyType(KeyType.HASH)
            .build());
        createDateKeySchema.add(KeySchemaElement.builder()
            .attributeName(ISSUE_ID_ATTR)
            .keyType(KeyType.RANGE)
            .build());
```

```
            final Projection createDateProjection = Projection.builder()
                .projectionType(ProjectionType.INCLUDE)
                .nonKeyAttributes(DESCRIPTION_ATTR, STATUS_ATTR)
                .build();

            final GlobalSecondaryIndex createDateIndex =
GlobalSecondaryIndex.builder()
                .indexName(CREATE_DATE_INDEX)
                .keySchema(createDateKeySchema)
                .projection(createDateProjection)
                .provisionedThroughput(ptIndex)
                .build();

            // TitleIndex
            final List<KeySchemaElement> titleKeySchema = new ArrayList<>();
            titleKeySchema.add(KeySchemaElement.builder()
                .attributeName(TITLE_ATTR)
                .keyType(KeyType.HASH)
                .build());
            titleKeySchema.add(KeySchemaElement.builder()
                .attributeName(ISSUE_ID_ATTR)
                .keyType(KeyType.RANGE)
                .build());

            final Projection titleProjection =

Projection.builder().projectionType(ProjectionType.KEYS_ONLY).build();

            final GlobalSecondaryIndex titleIndex = GlobalSecondaryIndex.builder()
                .indexName(TITLE_INDEX)
                .keySchema(titleKeySchema)
                .projection(titleProjection)
                .provisionedThroughput(ptIndex)
                .build();

            // DueDateIndex
            final List<KeySchemaElement> dueDateKeySchema = new ArrayList<>();
            dueDateKeySchema.add(KeySchemaElement.builder()
                .attributeName(DUE_DATE_ATTR)
                .keyType(KeyType.HASH)
                .build());

            final Projection dueDateProjection =
```

```
                Projection.builder().projectionType(ProjectionType.ALL).build();

            final GlobalSecondaryIndex dueDateIndex = GlobalSecondaryIndex.builder()
                .indexName(DUE_DATE_INDEX)
                .keySchema(dueDateKeySchema)
                .projection(dueDateProjection)
                .provisionedThroughput(ptIndex)
                .build();

            final CreateTableRequest createTableRequest =
CreateTableRequest.builder()
                .tableName(TABLE_NAME)
                .keySchema(tableKeySchema)
                .attributeDefinitions(attributeDefinitions)
                .globalSecondaryIndexes(createDateIndex, titleIndex, dueDateIndex)
                .provisionedThroughput(ProvisionedThroughput.builder()
                    .readCapacityUnits(1L)
                    .writeCapacityUnits(1L)
                    .build())
                .build();

            System.out.println("Creating table " + TABLE_NAME + "...");
            dynamoDbClient.createTable(createTableRequest);

            // Wait for table to become active
            System.out.println("Waiting for " + TABLE_NAME + " to become
ACTIVE...");
            final DynamoDbWaiter waiter = dynamoDbClient.waiter();
            final DescribeTableRequest describeTableRequest =
                DescribeTableRequest.builder().tableName(TABLE_NAME).build();

            final WaiterResponse<DescribeTableResponse> waiterResponse =
                waiter.waitUntilTableExists(describeTableRequest);
            waiterResponse.matched().response().ifPresent(response ->
System.out.println("Table is now ready for use"));

        } catch (DynamoDbException e) {
            System.err.println("Error creating table: " + e.getMessage());
            e.printStackTrace();
        }
    }
```

- Per i dettagli sull'API, consulta la sezione AWS SDK for Java 2.x API CreateTableReference.

Crea una tabella con la velocità effettiva a caldo abilitata

Il seguente esempio di codice mostra come creare una tabella con la velocità effettiva calda abilitata.

SDK per Java 2.x

Crea una tabella DynamoDB con l'impostazione della velocità effettiva calda utilizzando. AWS SDK for Java 2.x

```
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.GlobalSecondaryIndex;
import software.amazon.awssdk.services.dynamodb.model.KeySchemaElement;
import software.amazon.awssdk.services.dynamodb.model.KeyType;
import software.amazon.awssdk.services.dynamodb.model.Projection;
import software.amazon.awssdk.services.dynamodb.model.ProvisionedThroughput;
import software.amazon.awssdk.services.dynamodb.model.ScalarAttributeType;
import software.amazon.awssdk.services.dynamodb.model.WarmThroughput;

    public static WarmThroughput buildWarmThroughput(final Long readUnitsPerSecond,
 final Long writeUnitsPerSecond) {
        return WarmThroughput.builder()
            .readUnitsPerSecond(readUnitsPerSecond)
            .writeUnitsPerSecond(writeUnitsPerSecond)
            .build();
    }

    /**
     * Builds a ProvisionedThroughput object with the specified read and write
 capacity units.
     *
     * @param readCapacityUnits The read capacity units
     * @param writeCapacityUnits The write capacity units
     * @return A configured ProvisionedThroughput object
     */
    public static ProvisionedThroughput buildProvisionedThroughput(
        final Long readCapacityUnits, final Long writeCapacityUnits) {
        return ProvisionedThroughput.builder()
            .readCapacityUnits(readCapacityUnits)
            .writeCapacityUnits(writeCapacityUnits)
            .build();
    }
```

```
    /**
     * Builds an AttributeDefinition with the specified name and type.
     *
     * @param attributeName The attribute name
     * @param scalarAttributeType The attribute type
     * @return A configured AttributeDefinition
     */
    private static AttributeDefinition buildAttributeDefinition(
        final String attributeName, final ScalarAttributeType scalarAttributeType) {
        return AttributeDefinition.builder()
            .attributeName(attributeName)
            .attributeType(scalarAttributeType)
            .build();
    }

    /**
     * Builds a KeySchemaElement with the specified name and key type.
     *
     * @param attributeName The attribute name
     * @param keyType The key type (HASH or RANGE)
     * @return A configured KeySchemaElement
     */
    private static KeySchemaElement buildKeySchemaElement(final String
attributeName, final KeyType keyType) {
        return KeySchemaElement.builder()
            .attributeName(attributeName)
            .keyType(keyType)
            .build();
    }

    /**
     * Creates a DynamoDB table with the specified configuration including warm
throughput settings.
     *
     * @param ddb The DynamoDB client
     * @param tableName The name of the table to create
     * @param partitionKey The partition key attribute name
     * @param sortKey The sort key attribute name
     * @param miscellaneousKeyAttribute Additional key attribute name for GSI
     * @param nonKeyAttribute Non-key attribute to include in GSI projection
     * @param tableReadCapacityUnits Read capacity units for the table
     * @param tableWriteCapacityUnits Write capacity units for the table
     * @param tableWarmReadUnitsPerSecond Warm read units per second for the table
```

```
    * @param tableWarmWriteUnitsPerSecond Warm write units per second for the table
    * @param globalSecondaryIndexName The name of the GSI to create
    * @param globalSecondaryIndexReadCapacityUnits Read capacity units for the GSI
    * @param globalSecondaryIndexWriteCapacityUnits Write capacity units for the
GSI
    * @param globalSecondaryIndexWarmReadUnitsPerSecond Warm read units per second
for the GSI
    * @param globalSecondaryIndexWarmWriteUnitsPerSecond Warm write units per
second for the GSI
    */
   public static void createDynamoDBTable(
       final DynamoDbClient ddb,
       final String tableName,
       final String partitionKey,
       final String sortKey,
       final String miscellaneousKeyAttribute,
       final String nonKeyAttribute,
       final Long tableReadCapacityUnits,
       final Long tableWriteCapacityUnits,
       final Long tableWarmReadUnitsPerSecond,
       final Long tableWarmWriteUnitsPerSecond,
       final String globalSecondaryIndexName,
       final Long globalSecondaryIndexReadCapacityUnits,
       final Long globalSecondaryIndexWriteCapacityUnits,
       final Long globalSecondaryIndexWarmReadUnitsPerSecond,
       final Long globalSecondaryIndexWarmWriteUnitsPerSecond) {

       // Define the table attributes
       final AttributeDefinition partitionKeyAttribute =
buildAttributeDefinition(partitionKey, ScalarAttributeType.S);
       final AttributeDefinition sortKeyAttribute =
buildAttributeDefinition(sortKey, ScalarAttributeType.S);
       final AttributeDefinition miscellaneousKeyAttributeDefinition =
           buildAttributeDefinition(miscellaneousKeyAttribute,
ScalarAttributeType.N);
       final AttributeDefinition[] attributeDefinitions = {
           partitionKeyAttribute, sortKeyAttribute,
miscellaneousKeyAttributeDefinition
       };

       // Define the table key schema
       final KeySchemaElement partitionKeyElement =
buildKeySchemaElement(partitionKey, KeyType.HASH);
```

```
        final KeySchemaElement sortKeyElement = buildKeySchemaElement(sortKey,
KeyType.RANGE);
        final KeySchemaElement[] keySchema = {partitionKeyElement, sortKeyElement};

        // Define the provisioned throughput for the table
        final ProvisionedThroughput provisionedThroughput =
            buildProvisionedThroughput(tableReadCapacityUnits,
tableWriteCapacityUnits);

        // Define the Global Secondary Index (GSI)
        final KeySchemaElement globalSecondaryIndexPartitionKeyElement =
buildKeySchemaElement(sortKey, KeyType.HASH);
        final KeySchemaElement globalSecondaryIndexSortKeyElement =
            buildKeySchemaElement(miscellaneousKeyAttribute, KeyType.RANGE);
        final KeySchemaElement[] gsiKeySchema = {
            globalSecondaryIndexPartitionKeyElement,
globalSecondaryIndexSortKeyElement
        };

        final Projection gsiProjection = Projection.builder()
            .projectionType(PROJECTION_TYPE_INCLUDE)
            .nonKeyAttributes(nonKeyAttribute)
            .build();

        final ProvisionedThroughput gsiProvisionedThroughput =
            buildProvisionedThroughput(globalSecondaryIndexReadCapacityUnits,
globalSecondaryIndexWriteCapacityUnits);

        // Define the warm throughput for the Global Secondary Index (GSI)
        final WarmThroughput gsiWarmThroughput = buildWarmThroughput(
            globalSecondaryIndexWarmReadUnitsPerSecond,
globalSecondaryIndexWarmWriteUnitsPerSecond);

        final GlobalSecondaryIndex globalSecondaryIndex =
GlobalSecondaryIndex.builder()
            .indexName(globalSecondaryIndexName)
            .keySchema(gsiKeySchema)
            .projection(gsiProjection)
            .provisionedThroughput(gsiProvisionedThroughput)
            .warmThroughput(gsiWarmThroughput)
            .build();

        // Define the warm throughput for the table
        final WarmThroughput tableWarmThroughput =
```

```
            buildWarmThroughput(tableWarmReadUnitsPerSecond,
tableWarmWriteUnitsPerSecond);

        final CreateTableRequest request = CreateTableRequest.builder()
            .tableName(tableName)
            .attributeDefinitions(attributeDefinitions)
            .keySchema(keySchema)
            .provisionedThroughput(provisionedThroughput)
            .globalSecondaryIndexes(globalSecondaryIndex)
            .warmThroughput(tableWarmThroughput)
            .build();

        final CreateTableResponse response = ddb.createTable(request);
        System.out.println(response);
    }
```

- Per i dettagli sull'API, consulta la sezione API CreateTableReference AWS SDK for Java 2.x .

Creazione di un'applicazione Web per tracciare i dati DynamoDB

Il seguente esempio di codice mostra come creare un'applicazione Web che tiene traccia degli elementi di lavoro in una tabella Amazon DynamoDB e utilizza Amazon Simple Email Service (Amazon SES) per inviare report.

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, consulta l'esempio completo su. GitHub

Servizi utilizzati in questo esempio

- DynamoDB
- Amazon SES

Crea un elemento con un TTL

Il seguente esempio di codice mostra come creare un elemento con TTL.

SDK per Java 2.x

```
package com.amazon.samplelib.ttl;

import com.amazon.samplelib.CodeSampleUtils;
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;
import java.util.Map;
import java.util.Optional;

/**
 * Creates an item in a DynamoDB table with TTL attributes.
 * This class demonstrates how to add TTL expiration timestamps to DynamoDB items.
 */
public class CreateTTL {

    private static 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)
            """;
    private static final int DAYS_TO_EXPIRE = 90;
    private static final int SECONDS_PER_DAY = 24 * 60 * 60;
    private static final String PRIMARY_KEY_ATTR = "primaryKey";
    private static final String SORT_KEY_ATTR = "sortKey";
    private static final String CREATION_DATE_ATTR = "creationDate";
    private static final String EXPIRE_AT_ATTR = "expireAt";
```

```
   private static final String SUCCESS_MESSAGE = "%s PutItem operation with TTL
successful.";
   private static final String TABLE_NOT_FOUND_ERROR = "Error: The Amazon DynamoDB
table \"%s\" can't be found.";

   private final DynamoDbClient dynamoDbClient;

   /**
    * Constructs a CreateTTL instance with the specified DynamoDB client.
    *
    * @param dynamoDbClient The DynamoDB client to use
    */
   public CreateTTL(final DynamoDbClient dynamoDbClient) {
       this.dynamoDbClient = dynamoDbClient;
   }

   /**
    * Constructs a CreateTTL with a default DynamoDB client.
    */
   public CreateTTL() {
       this.dynamoDbClient = null;
   }

   /**
    * Main method to demonstrate creating an item with TTL.
    *
    * @param args Command line arguments
    */
   public static void main(final String[] args) {
       try {
           int result = new CreateTTL().processArgs(args);
           System.exit(result);
       } catch (Exception e) {
           System.err.println(e.getMessage());
           System.exit(1);
       }
   }

   /**
    * Process command line arguments and create an item with TTL.
    *
    * @param args Command line arguments
    * @return 0 if successful, non-zero otherwise
    * @throws ResourceNotFoundException If the table doesn't exist
```

```
     * @throws DynamoDbException If an error occurs during the operation
     * @throws IllegalArgumentException If arguments are invalid
     */
    public int processArgs(final String[] args) {
        // Argument validation (remove or replace this line when reusing this code)
        CodeSampleUtils.validateArgs(args, new int[] {3, 4}, USAGE);

        final String tableName = args[0];
        final String primaryKey = args[1];
        final String sortKey = args[2];
        final Region region = Optional.ofNullable(args.length > 3 ? args[3] : null)
            .map(Region::of)
            .orElse(Region.US_EAST_1);

        try (DynamoDbClient ddb = dynamoDbClient != null
            ? dynamoDbClient
            : DynamoDbClient.builder().region(region).build()) {
            final CreateTTL createTTL = new CreateTTL(ddb);
            createTTL.createItemWithTTL(tableName, primaryKey, sortKey);
            return 0;
        } catch (Exception e) {
            throw e;
        }
    }

    /**
     * Creates an item in the specified table with TTL attributes.
     *
     * @param tableName The name of the table
     * @param primaryKeyValue The value for the primary key
     * @param sortKeyValue The value for the sort key
     * @return The response from the PutItem operation
     * @throws ResourceNotFoundException If the table doesn't exist
     * @throws DynamoDbException If an error occurs during the operation
     */
    public PutItemResponse createItemWithTTL(
        final String tableName, final String primaryKeyValue, final String
sortKeyValue) {
        // 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 + (DAYS_TO_EXPIRE * SECONDS_PER_DAY);
```

```
        final Map<String, AttributeValue> itemMap = new HashMap<>();
        itemMap.put(
            PRIMARY_KEY_ATTR, AttributeValue.builder().s(primaryKeyValue).build());
        itemMap.put(SORT_KEY_ATTR,
 AttributeValue.builder().s(sortKeyValue).build());
        itemMap.put(
            CREATION_DATE_ATTR,
            AttributeValue.builder().n(String.valueOf(createDate)).build());
        itemMap.put(
            EXPIRE_AT_ATTR,
            AttributeValue.builder().n(String.valueOf(expireDate)).build());

        final PutItemRequest request =
            PutItemRequest.builder().tableName(tableName).item(itemMap).build();

        try {
            final PutItemResponse response = dynamoDbClient.putItem(request);
            System.out.println(String.format(SUCCESS_MESSAGE, tableName));
            return response;
        } catch (ResourceNotFoundException e) {
            System.err.format(TABLE_NOT_FOUND_ERROR, tableName);
            throw e;
        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            throw e;
        }
    }
}
```

- Per i dettagli sull'API, consulta PutItem AWS SDK for Java 2.xAPI Reference.

Rilevamento dei DPI nelle immagini

Il seguente esempio di codice mostra come creare un'app che utilizza Amazon Rekognition per rilevare i dispositivi di protezione individuale (DPI) nelle immagini.

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

Monitora le prestazioni di DynamoDB

Il seguente esempio di codice mostra come configurare l'uso di DynamoDB da parte di un'applicazione per monitorare le prestazioni.

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

Esegui operazioni di interrogazione avanzate

Il seguente esempio di codice mostra come eseguire operazioni di interrogazione avanzate in DynamoDB.

- Tabelle di interrogazione che utilizzano varie tecniche di filtraggio e condizione.
- Implementa l'impaginazione per set di risultati di grandi dimensioni.
- Usa gli indici secondari globali per modelli di accesso alternativi.
- Applica controlli di coerenza in base ai requisiti dell'applicazione.

SDK per Java 2.x

Esegui query con letture fortemente coerenti utilizzando AWS SDK for Java 2.x.

```
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 java.util.HashMap;
import java.util.Map;
import java.util.logging.Level;
import java.util.logging.Logger;

    public QueryResponse queryWithConsistentReads(
        final String tableName,
        final String partitionKeyName,
        final String partitionKeyValue,
        final boolean useConsistentRead) {

        CodeSampleUtils.validateTableParameters(tableName, partitionKeyName,
 partitionKeyValue);

        // Create expression attribute names for the column names
        final Map<String, String> expressionAttributeNames = new HashMap<>();
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_PK,
 partitionKeyName);

        // Create expression attribute values for the column values
        final Map<String, AttributeValue> expressionAttributeValues = new
 HashMap<>();
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_PK,
            AttributeValue.builder().s(partitionKeyValue).build());

        // Create the query request
        final QueryRequest queryRequest = QueryRequest.builder()
            .tableName(tableName)
            .keyConditionExpression(KEY_CONDITION_EXPRESSION)
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .consistentRead(useConsistentRead)
```

```
            .build();

        try {
            final QueryResponse response = dynamoDbClient.query(queryRequest);
            LOGGER.log(Level.INFO, "Query successful. Found {0} items",
 response.count());
            return response;
        } catch (ResourceNotFoundException e) {
            LOGGER.log(Level.SEVERE, "Table not found: {0}", tableName);
            throw e;
        } catch (DynamoDbException e) {
            LOGGER.log(Level.SEVERE, "Error querying with consistent reads", e);
            throw e;
        }
    }
```

Esegui una query utilizzando un indice secondario globale con AWS SDK for Java 2.x.

```
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 java.util.HashMap;
import java.util.Map;

    public QueryResponse queryTable(
        final String tableName, final String partitionKeyName, final String
 partitionKeyValue) {

        CodeSampleUtils.validateTableParameters(tableName, partitionKeyName,
 partitionKeyValue);

        // Create expression attribute names for the column names
        final Map<String, String> expressionAttributeNames = new HashMap<>();
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_PK,
 partitionKeyName);

        // Create expression attribute values for the column values
```

```
        final Map<String, AttributeValue> expressionAttributeValues = new
HashMap<>();
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_PK,
            AttributeValue.builder().s(partitionKeyValue).build());

        // Create the query request
        final QueryRequest queryRequest = QueryRequest.builder()
            .tableName(tableName)
            .keyConditionExpression(KEY_CONDITION_EXPRESSION)
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .build();

        try {
            final QueryResponse response = dynamoDbClient.query(queryRequest);
            System.out.println("Query on base table successful. Found " +
response.count() + " items");
            return response;
        } catch (ResourceNotFoundException e) {
            System.err.format("Error: The Amazon DynamoDB table \"%s\" can't be
found.\n", tableName);
            throw new DynamoDbQueryException("Table not found: " + tableName, e);
        } catch (DynamoDbException e) {
            System.err.println("Error querying base table: " + e.getMessage());
            throw new DynamoDbQueryException("Failed to execute query on base
table", e);
        }
    }

    /**
     * Queries a DynamoDB Global Secondary Index (GSI) by partition key.
     *
     * @param tableName         The name of the DynamoDB table
     * @param indexName         The name of the GSI
     * @param partitionKeyName  The name of the GSI partition key attribute
     * @param partitionKeyValue The value of the GSI partition key to query
     * @return The query response from DynamoDB
     * @throws ResourceNotFoundException if the table or index doesn't exist
     * @throws DynamoDbException if the query fails
     */
    public QueryResponse queryGlobalSecondaryIndex(
        final String tableName, final String indexName, final String
partitionKeyName, final String partitionKeyValue) {
```

```
        CodeSampleUtils.validateTableParameters(tableName, partitionKeyName,
partitionKeyValue);
        CodeSampleUtils.validateStringParameter("Index name", indexName);

        // Create expression attribute names for the column names
        final Map<String, String> expressionAttributeNames = new HashMap<>();
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_IK,
partitionKeyName);

        // Create expression attribute values for the column values
        final Map<String, AttributeValue> expressionAttributeValues = new
HashMap<>();
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_IK,
            AttributeValue.builder().s(partitionKeyValue).build());

        // Create the query request
        final QueryRequest queryRequest = QueryRequest.builder()
            .tableName(tableName)
            .indexName(indexName)
            .keyConditionExpression(GSI_KEY_CONDITION_EXPRESSION)
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .build();

        try {
            final QueryResponse response = dynamoDbClient.query(queryRequest);
            System.out.println("Query on GSI successful. Found " + response.count()
+ " items");
            return response;
        } catch (ResourceNotFoundException e) {
            System.err.format(
                "Error: The Amazon DynamoDB table \"%s\" or index \"%s\" can't be
found.\n", tableName, indexName);
            throw new DynamoDbQueryException("Table or index not found: " +
tableName + "/" + indexName, e);
        } catch (DynamoDbException e) {
            System.err.println("Error querying GSI: " + e.getMessage());
            throw new DynamoDbQueryException("Failed to execute query on GSI", e);
        }
    }
```

Interrogazione con impaginazione utilizzando AWS SDK for Java 2.x.

```
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 java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

    public List<Map<String, AttributeValue>> queryWithPagination(
        final String tableName, final String partitionKeyName, final String
 partitionKeyValue, final int pageSize) {

        CodeSampleUtils.validateTableParameters(tableName, partitionKeyName,
 partitionKeyValue);
        CodeSampleUtils.validatePositiveInteger("Page size", pageSize);

        // Create expression attribute names for the column names
        final Map<String, String> expressionAttributeNames = new HashMap<>();
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_PK,
 partitionKeyName);

        // Create expression attribute values for the column values
        final Map<String, AttributeValue> expressionAttributeValues = new
 HashMap<>();
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_PK,
            AttributeValue.builder().s(partitionKeyValue).build());

        // Create the query request
        QueryRequest.Builder queryRequestBuilder = QueryRequest.builder()
            .tableName(tableName)
            .keyConditionExpression(KEY_CONDITION_EXPRESSION)
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .limit(pageSize);

        // List to store all items from all pages
```

```
        final List<Map<String, AttributeValue>> allItems = new ArrayList<>();

        // Map to store the last evaluated key for pagination
        Map<String, AttributeValue> lastEvaluatedKey = null;
        int pageNumber = 1;

        try {
            do {
                // If we have a last evaluated key, use it for the next page
                if (lastEvaluatedKey != null) {
                    queryRequestBuilder.exclusiveStartKey(lastEvaluatedKey);
                }

                // Execute the query
                final QueryResponse response =
dynamoDbClient.query(queryRequestBuilder.build());

                // Process the current page of results
                final List<Map<String, AttributeValue>> pageItems =
response.items();
                allItems.addAll(pageItems);

                // Get the last evaluated key for the next page
                lastEvaluatedKey = response.lastEvaluatedKey();
                if (lastEvaluatedKey != null && lastEvaluatedKey.isEmpty()) {
                    lastEvaluatedKey = null;
                }

                System.out.println("Page " + pageNumber + ": Retrieved " +
pageItems.size() + " items (Running total: "
                    + allItems.size() + ")");

                pageNumber++;

            } while (lastEvaluatedKey != null);

            System.out.println("Query with pagination complete. Retrieved a total of
" + allItems.size()
                + " items across " + (pageNumber - 1) + " pages");

            return allItems;
        } catch (ResourceNotFoundException e) {
            System.err.format("Error: The Amazon DynamoDB table \"%s\" can't be
found.\n", tableName);
```

```
            throw e;
        } catch (DynamoDbException e) {
            System.err.println("Error querying with pagination: " + e.getMessage());
            throw e;
        }
    }
```

Interrogazione con filtri complessi utilizzando AWS SDK for Java 2.x.

```
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 java.util.HashMap;
import java.util.Map;
import java.util.logging.Level;
import java.util.logging.Logger;

    public QueryResponse queryWithComplexFilter(
        final String tableName,
        final String partitionKeyName,
        final String partitionKeyValue,
        final String statusAttrName,
        final String activeStatus,
        final String pendingStatus,
        final String priceAttrName,
        final double minPrice,
        final double maxPrice,
        final String categoryAttrName) {

        // Validate parameters
        CodeSampleUtils.validateTableParameters(tableName, partitionKeyName,
 partitionKeyValue);
        CodeSampleUtils.validateStringParameter("Status attribute name",
 statusAttrName);
        CodeSampleUtils.validateStringParameter("Active status", activeStatus);
        CodeSampleUtils.validateStringParameter("Pending status", pendingStatus);
```

```
        CodeSampleUtils.validateStringParameter("Price attribute name",
priceAttrName);
        CodeSampleUtils.validateStringParameter("Category attribute name",
categoryAttrName);
        CodeSampleUtils.validateNumericRange("Minimum price", minPrice, 0.0,
Double.MAX_VALUE);
        CodeSampleUtils.validateNumericRange("Maximum price", maxPrice, minPrice,
Double.MAX_VALUE);

        // Create expression attribute names for the column names
        final Map<String, String> expressionAttributeNames = new HashMap<>();
        expressionAttributeNames.put("#pk", partitionKeyName);
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_STATUS,
statusAttrName);
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_PRICE,
priceAttrName);
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_CATEGORY,
categoryAttrName);

        // Create expression attribute values for the column values
        final Map<String, AttributeValue> expressionAttributeValues = new
HashMap<>();
        expressionAttributeValues.put(
            ":pkValue", AttributeValue.builder().s(partitionKeyValue).build());
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_ACTIVE,
            AttributeValue.builder().s(activeStatus).build());
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_PENDING,
            AttributeValue.builder().s(pendingStatus).build());
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_MIN_PRICE,
            AttributeValue.builder().n(String.valueOf(minPrice)).build());
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_MAX_PRICE,
            AttributeValue.builder().n(String.valueOf(maxPrice)).build());

        // Create the query request
        final QueryRequest queryRequest = QueryRequest.builder()
            .tableName(tableName)
            .keyConditionExpression(KEY_CONDITION_EXPRESSION)
            .filterExpression(FILTER_EXPRESSION)
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
```

```
            .build();

        return dynamoDbClient.query(queryRequest);
    }
```

Interrogazione con un'espressione di filtro costruita dinamicamente utilizzando AWS SDK for Java 2.x.

```
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 java.util.HashMap;
import java.util.Map;

    public static QueryResponse queryWithDynamicFilter(
        final String tableName,
        final String partitionKeyName,
        final String partitionKeyValue,
        final Map<String, Object> filterCriteria,
        final Region region,
        final DynamoDbClient dynamoDbClient) {

        validateParameters(tableName, partitionKeyName, partitionKeyValue,
 filterCriteria);

        DynamoDbClient ddbClient = dynamoDbClient;
        boolean shouldClose = false;

        try {
            if (ddbClient == null) {
                ddbClient = createClient(region);
                shouldClose = true;
            }

            final QueryWithDynamicFilter queryHelper = new
 QueryWithDynamicFilter(ddbClient);
```

```
            return queryHelper.queryWithDynamicFilter(tableName, partitionKeyName,
partitionKeyValue, filterCriteria);
        } catch (ResourceNotFoundException e) {
            System.err.println("Table not found: " + tableName);
            throw e;
        } catch (DynamoDbException e) {
            System.err.println("Failed to execute dynamic filter query: " +
e.getMessage());
            throw e;
        } catch (Exception e) {
            System.err.println("Unexpected error during query: " + e.getMessage());
            throw e;
        } finally {
            if (shouldClose && ddbClient != null) {
                ddbClient.close();
            }
        }
    }

    public static void main(String[] args) {
        final String usage =
            """
                Usage:
                    <tableName> <partitionKeyName> <partitionKeyValue>
<filterAttrName> <filterAttrValue> [region]
                Where:
                    tableName - The Amazon DynamoDB table to query.
                    partitionKeyName - The name of the partition key attribute.
                    partitionKeyValue - The value of the partition key to query.
                    filterAttrName - The name of the attribute to filter on.
                    filterAttrValue - The value to filter by.
                    region (optional) - The AWS region where the table exists.
(Default: us-east-1)
                """;

        if (args.length < 5) {
            System.out.println(usage);
            System.exit(1);
        }

        final String tableName = args[0];
        final String partitionKeyName = args[1];
        final String partitionKeyValue = args[2];
        final String filterAttrName = args[3];
```

```
        final String filterAttrValue = args[4];
        final Region region = args.length > 5 ? Region.of(args[5]) :
Region.US_EAST_1;

        System.out.println("Querying items with dynamic filter: " + filterAttrName +
" = " + filterAttrValue);

        try {
            // Using the builder pattern to create and execute the query
            final QueryResponse response = new DynamicFilterQueryBuilder()
                .withTableName(tableName)
                .withPartitionKeyName(partitionKeyName)
                .withPartitionKeyValue(partitionKeyValue)
                .withFilterCriterion(filterAttrName, filterAttrValue)
                .withRegion(region)
                .execute();

            // Process the results
            System.out.println("Found " + response.count() + " items:");
            response.items().forEach(item -> System.out.println(item));

            // Demonstrate multiple filter criteria
            System.out.println("\nNow querying with multiple filter criteria:");

            Map<String, Object> multipleFilters = new HashMap<>();
            multipleFilters.put(filterAttrName, filterAttrValue);
            multipleFilters.put("status", "active");

            final QueryResponse multiFilterResponse = new
DynamicFilterQueryBuilder()
                .withTableName(tableName)
                .withPartitionKeyName(partitionKeyName)
                .withPartitionKeyValue(partitionKeyValue)
                .withFilterCriteria(multipleFilters)
                .withRegion(region)
                .execute();

            System.out.println("Found " + multiFilterResponse.count() + " items with
multiple filters:");
            multiFilterResponse.items().forEach(item -> System.out.println(item));

        } catch (IllegalArgumentException e) {
            System.err.println("Invalid input: " + e.getMessage());
            System.exit(1);
```

```
        } catch (ResourceNotFoundException e) {
            System.err.println("Table not found: " + tableName);
            System.exit(1);
        } catch (DynamoDbException e) {
            System.err.println("DynamoDB error: " + e.getMessage());
            System.exit(1);
        } catch (Exception e) {
            System.err.println("Unexpected error: " + e.getMessage());
            System.exit(1);
        }
    }
```

Interroga con un'espressione di filtro e limita l'utilizzo AWS SDK for Java 2.x.

```
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 java.util.HashMap;
import java.util.Map;
import java.util.logging.Level;
import java.util.logging.Logger;

    public QueryResponse queryWithFilterAndLimit(
        final String tableName,
        final String partitionKeyName,
        final String partitionKeyValue,
        final String filterAttrName,
        final String filterAttrValue,
        final int limit) {

        CodeSampleUtils.validateTableParameters(tableName, partitionKeyName,
 partitionKeyValue);
        CodeSampleUtils.validateStringParameter("Filter attribute name",
 filterAttrName);
        CodeSampleUtils.validateStringParameter("Filter attribute value",
 filterAttrValue);
        CodeSampleUtils.validatePositiveInteger("Limit", limit);
```

```
        // Create expression attribute names for the column names
        final Map<String, String> expressionAttributeNames = new HashMap<>();
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_PK,
partitionKeyName);
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_FILTER,
filterAttrName);

        // Create expression attribute values for the column values
        final Map<String, AttributeValue> expressionAttributeValues = new
HashMap<>();
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_PK,
            AttributeValue.builder().s(partitionKeyValue).build());
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_FILTER,
            AttributeValue.builder().s(filterAttrValue).build());

        // Create the filter expression
        final String filterExpression = "#filterAttr = :filterValue";

        // Create the query request
        final QueryRequest queryRequest = QueryRequest.builder()
            .tableName(tableName)
            .keyConditionExpression(KEY_CONDITION_EXPRESSION)
            .filterExpression(filterExpression)
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .limit(limit)
            .build();

        try {
            final QueryResponse response = dynamoDbClient.query(queryRequest);
            LOGGER.log(Level.INFO, "Query with filter and limit successful. Found
{0} items", response.count());
            LOGGER.log(
                Level.INFO, "ScannedCount: {0} (total items evaluated before
filtering)", response.scannedCount());
            return response;
        } catch (ResourceNotFoundException e) {
            LOGGER.log(Level.SEVERE, "Table not found: {0}", tableName);
            throw e;
        } catch (DynamoDbException e) {
```

```
            LOGGER.log(Level.SEVERE, "Error querying with filter and limit: {0}",
e.getMessage());
            throw e;
        }
    }
```

- Per ulteriori informazioni sulle API, consulta Query nella Documentazione di riferimento delle API AWS SDK for Java 2.x .

Esegui operazioni sull'elenco

Il seguente esempio di codice mostra come eseguire operazioni sugli elenchi in DynamoDB.

- Aggiungere elementi a un attributo di elenco.
- Rimuove elementi da un attributo di elenco.
- Aggiorna elementi specifici in un elenco per indice.
- Usa le funzioni di aggiunta alla lista e all'indice delle liste.

SDK per Java 2.x

Dimostra le operazioni sugli elenchi usando AWS SDK for Java 2.x.

```
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.GetItemRequest;
import software.amazon.awssdk.services.dynamodb.model.GetItemResponse;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemRequest;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemResponse;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

    /**
     * Appends items to a list attribute.
     *
     * <p>This method demonstrates how to use the list_append function to add
     * items to the end of a list attribute.
```

```
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param listAttributeName The name of the list attribute
     * @param itemsToAppend The items to append to the list
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static UpdateItemResponse appendToList(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        String listAttributeName,
        List<AttributeValue> itemsToAppend) {

        // Create a list value from the items to append
        AttributeValue listValue =
AttributeValue.builder().l(itemsToAppend).build();

        // Define the update parameters
        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .updateExpression("SET #attrName =
list_append(if_not_exists(#attrName, :emptyList), :newItems)")
            .expressionAttributeNames(Map.of("#attrName", listAttributeName))
            .expressionAttributeValues(Map.of(
                ":newItems",
                listValue,
                ":emptyList",
                AttributeValue.builder().l(new
ArrayList<AttributeValue>()).build()))
            .returnValues("UPDATED_NEW")
            .build();

        // Perform the update operation
        return dynamoDbClient.updateItem(request);
    }

    /**
     * Prepends items to a list attribute.
     *
     * <p>This method demonstrates how to use the list_append function to add
```

```
     * items to the beginning of a list attribute.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param listAttributeName The name of the list attribute
     * @param itemsToPrepend The items to prepend to the list
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static UpdateItemResponse prependToList(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        String listAttributeName,
        List<AttributeValue> itemsToPrepend) {

        // Create a list value from the items to prepend
        AttributeValue listValue =
AttributeValue.builder().l(itemsToPrepend).build();

        // Define the update parameters
        // Note: To prepend, we put the new items first in the list_append function
        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .updateExpression("SET #attrName = list_append(:newItems,
if_not_exists(#attrName, :emptyList))")
            .expressionAttributeNames(Map.of("#attrName", listAttributeName))
            .expressionAttributeValues(Map.of(
                ":newItems",
                listValue,
                ":emptyList",
                AttributeValue.builder().l(new
ArrayList<AttributeValue>()).build()))
            .returnValues("UPDATED_NEW")
            .build();

        // Perform the update operation
        return dynamoDbClient.updateItem(request);
    }

    /**
     * Updates a specific element in a list attribute.
```

```
     *
     * <p>This method demonstrates how to update a specific element in a list
     * by its index.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param listAttributeName The name of the list attribute
     * @param index The index of the element to update
     * @param newValue The new value for the element
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static UpdateItemResponse updateListElement(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        String listAttributeName,
        int index,
        AttributeValue newValue) {

        // Define the update parameters
        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .updateExpression("SET #attrName[" + index + "] = :newValue")
            .expressionAttributeNames(Map.of("#attrName", listAttributeName))
            .expressionAttributeValues(Map.of(":newValue", newValue))
            .returnValues("UPDATED_NEW")
            .build();

        // Perform the update operation
        return dynamoDbClient.updateItem(request);
    }

    /**
     * Removes a specific element from a list attribute.
     *
     * <p>This method demonstrates how to remove a specific element from a list
     * by its index.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
```

```
     * @param listAttributeName The name of the list attribute
     * @param index The index of the element to remove
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static UpdateItemResponse removeListElement(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        String listAttributeName,
        int index) {

        // Define the update parameters
        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .updateExpression("REMOVE #attrName[" + index + "]")
            .expressionAttributeNames(Map.of("#attrName", listAttributeName))
            .returnValues("UPDATED_NEW")
            .build();

        // Perform the update operation
        return dynamoDbClient.updateItem(request);
    }

    /**
     * Gets the current value of a list attribute.
     *
     * <p>Helper method to retrieve the current value of a list attribute.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to get
     * @param listAttributeName The name of the list attribute
     * @return The list attribute value or null if not found
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static List<AttributeValue> getListAttribute(
        DynamoDbClient dynamoDbClient, String tableName, Map<String, AttributeValue>
key, String listAttributeName) {

        // Define the get parameters
        GetItemRequest request = GetItemRequest.builder()
            .tableName(tableName)
```

```
            .key(key)
            .projectionExpression(listAttributeName)
            .build();

        try {
            // Perform the get operation
            GetItemResponse response = dynamoDbClient.getItem(request);

            // Return the list attribute if it exists, otherwise null
            if (response.item() != null &&
response.item().containsKey(listAttributeName)) {
                return response.item().get(listAttributeName).l();
            }

            return null;
        } catch (DynamoDbException e) {
            throw DynamoDbException.builder()
                .message("Failed to get list attribute: " + e.getMessage())
                .cause(e)
                .build();
        }
    }
```

Esempio di utilizzo delle operazioni di elenco con AWS SDK for Java 2.x.

```
    public static void exampleUsage(DynamoDbClient dynamoDbClient, String tableName)
{
        // Example key
        Map<String, AttributeValue> key = new HashMap<>();
        key.put("ProductId", AttributeValue.builder().s("P12345").build());

        System.out.println("Demonstrating list operations in DynamoDB");

        try {
            // Example 1: Append items to a list
            System.out.println("\nExample 1: Appending items to a list");
            List<AttributeValue> tagsToAppend = List.of(
                AttributeValue.builder().s("Electronics").build(),
                AttributeValue.builder().s("Gadget").build());

            UpdateItemResponse appendResponse = appendToList(dynamoDbClient,
tableName, key, "Tags", tagsToAppend);
```

```
            System.out.println("Updated list attribute: " +
appendResponse.attributes());

            // Example 2: Prepend items to a list
            System.out.println("\nExample 2: Prepending items to a list");
            List<AttributeValue> tagsToPrepend = List.of(
                AttributeValue.builder().s("Featured").build(),
                AttributeValue.builder().s("New").build());

            UpdateItemResponse prependResponse = prependToList(dynamoDbClient,
tableName, key, "Tags", tagsToPrepend);

            System.out.println("Updated list attribute: " +
prependResponse.attributes());

            // Example 3: Update a specific element in a list
            System.out.println("\nExample 3: Updating a specific element in a
list");
            UpdateItemResponse updateResponse = updateListElement(
                dynamoDbClient,
                tableName,
                key,
                "Tags",
                0,
                AttributeValue.builder().s("BestSeller").build());

            System.out.println("Updated list attribute: " +
updateResponse.attributes());

            // Example 4: Remove a specific element from a list
            System.out.println("\nExample 4: Removing a specific element from a
list");
            UpdateItemResponse removeResponse = removeListElement(dynamoDbClient,
tableName, key, "Tags", 1);

            System.out.println("Updated list attribute: " +
removeResponse.attributes());

            // Example 5: Get the current value of a list attribute
            System.out.println("\nExample 5: Getting the current value of a list
attribute");
            List<AttributeValue> currentList = getListAttribute(dynamoDbClient,
tableName, key, "Tags");
```

```
            if (currentList != null) {
                System.out.println("Current list attribute:");
                for (int i = 0; i < currentList.size(); i++) {
                    System.out.println("  [" + i + "]: " + currentList.get(i).s());
                }
            } else {
                System.out.println("List attribute not found");
            }

            // Explain list operations
            System.out.println("\nKey points about DynamoDB list operations:");
            System.out.println("1. Lists are ordered collections of attributes");
            System.out.println("2. Use list_append to add items to a list");
            System.out.println("3. To append items, use list_append(existingList,
newItems)");
            System.out.println("4. To prepend items, use list_append(newItems,
existingList)");
            System.out.println("5. Use index notation (list[0]) to access or update
specific elements");
            System.out.println("6. Use REMOVE to delete elements from a list");
            System.out.println("7. List indices are zero-based");
            System.out.println("8. Use if_not_exists to handle the case where the
list doesn't exist yet");

        } catch (DynamoDbException e) {
            System.err.println("Error: " + e.getMessage());
            e.printStackTrace();
        }
    }
```

- Per i dettagli sull'API, vedere UpdateItemin AWS SDK for Java 2.x API Reference.

Esegui operazioni sulla mappa

Il seguente esempio di codice mostra come eseguire operazioni di mappatura in DynamoDB.

- Aggiungi e aggiorna gli attributi annidati nelle strutture della mappa.
- Rimuovi campi specifici dalle mappe.
- Lavora con attributi di mappa profondamente annidati.

SDK per Java 2.x

Dimostra le operazioni sulla mappa utilizzando AWS SDK for Java 2.x.

```
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.GetItemRequest;
import software.amazon.awssdk.services.dynamodb.model.GetItemResponse;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemRequest;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemResponse;

import java.util.HashMap;
import java.util.List;
import java.util.Map;

    /**
     * Updates a map attribute that may not exist.
     *
     * <p>This method demonstrates how to safely update a map attribute
     * by using if_not_exists to handle the case where the map doesn't exist yet.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param mapName The name of the map attribute
     * @param mapKey The key within the map to update
     * @param value The value to set
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static UpdateItemResponse updateMapAttributeSafe(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        String mapName,
        String mapKey,
        AttributeValue value) {

        // Create an empty map to use if the map doesn't exist
        Map<String, AttributeValue> emptyMap = new HashMap<>();
        AttributeValue emptyMapValue = AttributeValue.builder().m(emptyMap).build();

        // Define the update parameters
```

```
        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .updateExpression("SET #mapName = if_not_exists(#mapName, :emptyMap),
#mapName.#mapKey = :value")
            .expressionAttributeNames(Map.of(
                "#mapName", mapName,
                "#mapKey", mapKey))
            .expressionAttributeValues(Map.of(
                ":value",
                value,
                ":emptyMap",
                AttributeValue.builder().m(new HashMap<>()).build()))
            .returnValues("UPDATED_NEW")
            .build();

        // Perform the update operation
        return dynamoDbClient.updateItem(request);
    }

    /**
     * Adds an attribute to a nested map.
     *
     * <p>This method demonstrates how to update a nested attribute without
     * overwriting the entire map.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param path The path to the nested attribute as a list
     * @param value The value to set
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static UpdateItemResponse addToNestedMap(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        List<String> path,
        AttributeValue value) {

        // Create expression attribute names for each part of the path
        Map<String, String> expressionAttributeNames = new HashMap<>();
        for (int i = 0; i < path.size(); i++) {
```

```
            expressionAttributeNames.put("#attr" + i, path.get(i));
        }

        // Build the attribute path using the expression attribute names
        StringBuilder attributePathExpression = new StringBuilder();
        for (int i = 0; i < path.size(); i++) {
            if (i > 0) {
                attributePathExpression.append(".");
            }
            attributePathExpression.append("#attr").append(i);
        }

        // Define the update parameters
        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .updateExpression("SET " + attributePathExpression.toString() + "
= :value")
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(Map.of(":value", value))
            .returnValues("UPDATED_NEW")
            .build();

        // Perform the update operation
        return dynamoDbClient.updateItem(request);
    }

    /**
     * Removes an attribute from a map.
     *
     * <p>This method demonstrates how to remove a specific attribute from a map.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param mapName The name of the map attribute
     * @param mapKey The key within the map to remove
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static UpdateItemResponse removeMapAttribute(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
```

```
        String mapName,
        String mapKey) {

        // Define the update parameters
        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .updateExpression("REMOVE #mapName.#mapKey")
            .expressionAttributeNames(Map.of(
                "#mapName", mapName,
                "#mapKey", mapKey))
            .returnValues("UPDATED_NEW")
            .build();

        // Perform the update operation
        return dynamoDbClient.updateItem(request);
    }

    /**
     * Creates a map with multiple attributes in a single operation.
     *
     * <p>This method demonstrates how to create a map with multiple attributes
     * in a single update operation.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param mapName The name of the map attribute
     * @param attributes The attributes to set in the map
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static UpdateItemResponse createMapWithAttributes(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        String mapName,
        Map<String, AttributeValue> attributes) {

        // Create a map value from the attributes
        AttributeValue mapValue = AttributeValue.builder().m(attributes).build();

        // Define the update parameters
        UpdateItemRequest request = UpdateItemRequest.builder()
```

```
            .tableName(tableName)
            .key(key)
            .updateExpression("SET #mapName = :mapValue")
            .expressionAttributeNames(Map.of("#mapName", mapName))
            .expressionAttributeValues(Map.of(":mapValue", mapValue))
            .returnValues("UPDATED_NEW")
            .build();

        // Perform the update operation
        return dynamoDbClient.updateItem(request);
    }

    /**
     * Gets the current value of a map attribute.
     *
     * <p>Helper method to retrieve the current value of a map attribute.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to get
     * @param mapName The name of the map attribute
     * @return The map attribute value or null if not found
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static Map<String, AttributeValue> getMapAttribute(
        DynamoDbClient dynamoDbClient, String tableName, Map<String, AttributeValue>
key, String mapName) {

        // Define the get parameters
        GetItemRequest request = GetItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .projectionExpression(mapName)
            .build();

        try {
            // Perform the get operation
            GetItemResponse response = dynamoDbClient.getItem(request);

            // Return the map attribute if it exists, otherwise null
            if (response.item() != null && response.item().containsKey(mapName)) {
                return response.item().get(mapName).m();
            }
```

```
            return null;
        } catch (DynamoDbException e) {
            throw DynamoDbException.builder()
                .message("Failed to get map attribute: " + e.getMessage())
                .cause(e)
                .build();
        }
    }
```

Esempio di utilizzo delle operazioni cartografiche con AWS SDK for Java 2.x.

```
    public static void exampleUsage(DynamoDbClient dynamoDbClient, String tableName)
{
        // Example key
        Map<String, AttributeValue> key = new HashMap<>();
        key.put("ProductId", AttributeValue.builder().s("P12345").build());

        System.out.println("Demonstrating map operations in DynamoDB");

        try {
            // Example 1: Create a map with multiple attributes
            System.out.println("\nExample 1: Creating a map with multiple
attributes");
            Map<String, AttributeValue> productDetails = new HashMap<>();
            productDetails.put("Color", AttributeValue.builder().s("Red").build());
            productDetails.put("Weight", AttributeValue.builder().n("2.5").build());
            productDetails.put(
                "Dimensions", AttributeValue.builder().s("10x20x5").build());

            UpdateItemResponse createResponse =
                createMapWithAttributes(dynamoDbClient, tableName, key, "Details",
productDetails);

            System.out.println("Created map attribute: " +
createResponse.attributes());

            // Example 2: Update a specific attribute in a map
            System.out.println("\nExample 2: Updating a specific attribute in a
map");
            UpdateItemResponse updateResponse = updateMapAttributeSafe(
                dynamoDbClient,
                tableName,
```

```
                key,
                "Details",
                "Color",
                AttributeValue.builder().s("Blue").build());

            System.out.println("Updated map attribute: " +
updateResponse.attributes());

            // Example 3: Add an attribute to a nested map
            System.out.println("\nExample 3: Adding an attribute to a nested map");
            UpdateItemResponse nestedResponse = addToNestedMap(
                dynamoDbClient,
                tableName,
                key,
                List.of("Specifications", "Technical", "Resolution"),
                AttributeValue.builder().s("1920x1080").build());

            System.out.println("Added to nested map: " +
nestedResponse.attributes());

            // Example 4: Remove an attribute from a map
            System.out.println("\nExample 4: Removing an attribute from a map");
            UpdateItemResponse removeResponse =
                removeMapAttribute(dynamoDbClient, tableName, key, "Details",
"Dimensions");

            System.out.println("Updated map after removal: " +
removeResponse.attributes());

            // Example 5: Get the current value of a map attribute
            System.out.println("\nExample 5: Getting the current value of a map
attribute");
            Map<String, AttributeValue> currentMap = getMapAttribute(dynamoDbClient,
tableName, key, "Details");

            if (currentMap != null) {
                System.out.println("Current map attribute:");
                for (Map.Entry<String, AttributeValue> entry :
currentMap.entrySet()) {
                    System.out.println("  " + entry.getKey() + ": " +
entry.getValue());
                }
            } else {
                System.out.println("Map attribute not found");
```

```
            }

            // Explain map operations
            System.out.println("\nKey points about DynamoDB map operations:");
            System.out.println("1. Maps are unordered collections of name-value
pairs");
            System.out.println("2. Use dot notation (map.key) to access or update
specific attributes");
            System.out.println("3. You can update individual attributes without
overwriting the entire map");
            System.out.println("4. Maps can be nested to create complex data
structures");
            System.out.println("5. Use REMOVE to delete attributes from a map");
            System.out.println("6. You can create a map with multiple attributes in
a single operation");
            System.out.println("7. Map keys are case-sensitive");

        } catch (DynamoDbException e) {
            System.err.println("Error: " + e.getMessage());
            e.printStackTrace();
        }
    }
```

- Per i dettagli sull'API, vedere UpdateItemin AWS SDK for Java 2.x API Reference.

Esegui operazioni impostate

Il seguente esempio di codice mostra come eseguire operazioni di set in DynamoDB.

- Aggiungere elementi a un attributo set.
- Rimuove elementi da un attributo set.
- Usa le operazioni ADD e DELETE con i set.

SDK per Java 2.x

Dimostra le operazioni sui set utilizzando AWS SDK for Java 2.x.

```
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.GetItemRequest;
import software.amazon.awssdk.services.dynamodb.model.GetItemResponse;
import software.amazon.awssdk.services.dynamodb.model.ReturnValue;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemRequest;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemResponse;

import java.util.HashMap;
import java.util.HashSet;
import java.util.Map;
import java.util.Set;

    /**
     * Adds values to a string set attribute.
     *
     * <p>This method demonstrates how to use the ADD operation to add values
     * to a string set attribute.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param setAttributeName The name of the set attribute
     * @param valuesToAdd The values to add to the set
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static UpdateItemResponse addToStringSet(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        String setAttributeName,
        Set<String> valuesToAdd) {

        // Create a string set value from the values to add
        AttributeValue setValue = AttributeValue.builder().ss(valuesToAdd).build();

        // Define the update parameters
        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .updateExpression("ADD #setAttr :valuesToAdd")
            .expressionAttributeNames(Map.of("#setAttr", setAttributeName))
            .expressionAttributeValues(Map.of(":valuesToAdd", setValue))
            .returnValues(ReturnValue.UPDATED_NEW)
            .build();
```

```
        // Perform the update operation
        return dynamoDbClient.updateItem(request);
    }

    /**
     * Adds values to a number set attribute.
     *
     * <p>This method demonstrates how to use the ADD operation to add values
     * to a number set attribute.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param setAttributeName The name of the set attribute
     * @param valuesToAdd The values to add to the set
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static UpdateItemResponse addToNumberSet(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        String setAttributeName,
        Set<Number> valuesToAdd) {

        // Convert numbers to strings for DynamoDB
        Set<String> stringValues = new HashSet<>();
        for (Number value : valuesToAdd) {
            stringValues.add(value.toString());
        }

        // Create a number set value from the values to add
        AttributeValue setValue = AttributeValue.builder().ns(stringValues).build();

        // Define the update parameters
        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .updateExpression("ADD #setAttr :valuesToAdd")
            .expressionAttributeNames(Map.of("#setAttr", setAttributeName))
            .expressionAttributeValues(Map.of(":valuesToAdd", setValue))
            .returnValues(ReturnValue.UPDATED_NEW)
            .build();
```

```
        // Perform the update operation
        return dynamoDbClient.updateItem(request);
    }

    /**
     * Removes values from a set attribute.
     *
     * <p>This method demonstrates how to use the DELETE operation to remove values
     * from a set attribute.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param setAttributeName The name of the set attribute
     * @param valuesToRemove The values to remove from the set
     * @param isNumberSet Whether the set is a number set (true) or string set
(false)
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static UpdateItemResponse removeFromSet(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        String setAttributeName,
        Set<?> valuesToRemove,
        boolean isNumberSet) {

        AttributeValue setValue;

        if (isNumberSet) {
            // Convert numbers to strings for DynamoDB
            Set<String> stringValues = new HashSet<>();
            for (Object value : valuesToRemove) {
                if (value instanceof Number) {
                    stringValues.add(value.toString());
                } else {
                    throw new IllegalArgumentException("Values must be numbers for a
number set");
                }
            }

            setValue = AttributeValue.builder().ns(stringValues).build();
```

```
        } else {
            // Convert objects to strings for DynamoDB
            Set<String> stringValues = new HashSet<>();
            for (Object value : valuesToRemove) {
                stringValues.add(value.toString());
            }

            setValue = AttributeValue.builder().ss(stringValues).build();
        }

        // Define the update parameters
        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .updateExpression("DELETE #setAttr :valuesToRemove")
            .expressionAttributeNames(Map.of("#setAttr", setAttributeName))
            .expressionAttributeValues(Map.of(":valuesToRemove", setValue))
            .returnValues(ReturnValue.UPDATED_NEW)
            .build();

        // Perform the update operation
        return dynamoDbClient.updateItem(request);
    }

    /**
     * Checks if a value exists in a set attribute.
     *
     * <p>This method demonstrates how to use the contains function to check
     * if a value exists in a set attribute.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to check
     * @param setAttributeName The name of the set attribute
     * @param valueToCheck The value to check for
     * @return Map containing the result of the check
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static Map<String, Object> checkIfValueInSet(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        String setAttributeName,
        String valueToCheck) {
```

```
        Map<String, Object> result = new HashMap<>();

        try {
            // Define the update parameters with a condition expression
            UpdateItemRequest request = UpdateItemRequest.builder()
                .tableName(tableName)
                .key(key)
                .updateExpression("SET #tempAttr = :tempVal")
                .conditionExpression("contains(#setAttr, :valueToCheck)")
                .expressionAttributeNames(Map.of("#setAttr", setAttributeName,
"#tempAttr", "TempAttribute"))
                .expressionAttributeValues(Map.of(
                    ":valueToCheck",
AttributeValue.builder().s(valueToCheck).build(),
                    ":tempVal", AttributeValue.builder().s("TempValue").build()))
                .returnValues(ReturnValue.UPDATED_NEW)
                .build();

            // Attempt the update operation
            dynamoDbClient.updateItem(request);

            // If we get here, the condition was met
            result.put("exists", true);
            result.put("message", "Value '" + valueToCheck + "' exists in the set");

            // Clean up the temporary attribute
            UpdateItemRequest cleanupRequest = UpdateItemRequest.builder()
                .tableName(tableName)
                .key(key)
                .updateExpression("REMOVE #tempAttr")
                .expressionAttributeNames(Map.of("#tempAttr", "TempAttribute"))
                .build();

            dynamoDbClient.updateItem(cleanupRequest);

        } catch (DynamoDbException e) {
            if (e.getMessage().contains("ConditionalCheckFailed")) {
                // The condition was not met
                result.put("exists", false);
                result.put("message", "Value '" + valueToCheck + "' does not exist
in the set");
            } else {
                // Some other error occurred
```

```
                result.put("exists", false);
                result.put("message", "Error checking set: " + e.getMessage());
                result.put("error", e.getClass().getSimpleName());
            }
        }

        return result;
    }

    /**
     * Creates a set with multiple values in a single operation.
     *
     * <p>This method demonstrates how to create a set with multiple values
     * in a single update operation.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param setAttributeName The name of the set attribute
     * @param setValues The values to include in the set
     * @param isNumberSet Whether to create a number set (true) or string set
(false)
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static UpdateItemResponse createSetWithValues(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        String setAttributeName,
        Set<?> setValues,
        boolean isNumberSet) {

        AttributeValue setValue;

        if (isNumberSet) {
            // Convert numbers to strings for DynamoDB
            Set<String> stringValues = new HashSet<>();
            for (Object value : setValues) {
                if (value instanceof Number) {
                    stringValues.add(value.toString());
                } else {
                    throw new IllegalArgumentException("Values must be numbers for a
number set");
```

```
                }
            }

            setValue = AttributeValue.builder().ns(stringValues).build();
        } else {
            // Convert objects to strings for DynamoDB
            Set<String> stringValues = new HashSet<>();
            for (Object value : setValues) {
                stringValues.add(value.toString());
            }

            setValue = AttributeValue.builder().ss(stringValues).build();
        }

        // Define the update parameters
        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .updateExpression("SET #setAttr = :setValue")
            .expressionAttributeNames(Map.of("#setAttr", setAttributeName))
            .expressionAttributeValues(Map.of(":setValue", setValue))
            .returnValues(ReturnValue.UPDATED_NEW)
            .build();

        // Perform the update operation
        return dynamoDbClient.updateItem(request);
    }

    /**
     * Gets the current value of a set attribute.
     *
     * <p>Helper method to retrieve the current value of a set attribute.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to get
     * @param setAttributeName The name of the set attribute
     * @return The set attribute value or null if not found
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static AttributeValue getSetAttribute(
        DynamoDbClient dynamoDbClient, String tableName, Map<String, AttributeValue>
key, String setAttributeName) {
```

```
        // Define the get parameters
        GetItemRequest request = GetItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .projectionExpression(setAttributeName)
            .build();

        try {
            // Perform the get operation
            GetItemResponse response = dynamoDbClient.getItem(request);

            // Return the set attribute if it exists, otherwise null
            if (response.item() != null &&
response.item().containsKey(setAttributeName)) {
                return response.item().get(setAttributeName);
            }

            return null;
        } catch (DynamoDbException e) {
            throw DynamoDbException.builder()
                .message("Failed to get set attribute: " + e.getMessage())
                .cause(e)
                .build();
        }
    }
```

Esempio di utilizzo delle operazioni di set con AWS SDK for Java 2.x.

```
    public static void exampleUsage(DynamoDbClient dynamoDbClient, String tableName)
{
        // Example key
        Map<String, AttributeValue> key = new HashMap<>();
        key.put("ProductId", AttributeValue.builder().s("P12345").build());

        System.out.println("Demonstrating set operations in DynamoDB");

        try {
            // Example 1: Create a string set with multiple values
            System.out.println("\nExample 1: Creating a string set with multiple
values");
            Set<String> tags = new HashSet<>();
            tags.add("Electronics");
```

```
            tags.add("Gadget");
            tags.add("Smartphone");

            UpdateItemResponse createResponse = createSetWithValues(
                dynamoDbClient, tableName, key, "Tags", tags, false // Not a number
set
                );

            System.out.println("Created set attribute: " +
createResponse.attributes());

            // Example 2: Add values to a string set
            System.out.println("\nExample 2: Adding values to a string set");
            Set<String> additionalTags = new HashSet<>();
            additionalTags.add("Mobile");
            additionalTags.add("Wireless");

            UpdateItemResponse addResponse = addToStringSet(dynamoDbClient,
tableName, key, "Tags", additionalTags);

            System.out.println("Updated set attribute: " +
addResponse.attributes());

            // Example 3: Create a number set with multiple values
            System.out.println("\nExample 3: Creating a number set with multiple
values");
            Set<Number> ratings = new HashSet<>();
            ratings.add(4);
            ratings.add(5);
            ratings.add(4.5);

            UpdateItemResponse createNumberSetResponse = createSetWithValues(
                dynamoDbClient, tableName, key, "Ratings", ratings, true // Is a
number set
                );

            System.out.println("Created number set attribute: " +
createNumberSetResponse.attributes());

            // Example 4: Add values to a number set
            System.out.println("\nExample 4: Adding values to a number set");
            Set<Number> additionalRatings = new HashSet<>();
            additionalRatings.add(3.5);
            additionalRatings.add(4.2);
```

```
            UpdateItemResponse addNumberResponse =
                addToNumberSet(dynamoDbClient, tableName, key, "Ratings",
additionalRatings);

            System.out.println("Updated number set attribute: " +
addNumberResponse.attributes());

            // Example 5: Remove values from a set
            System.out.println("\nExample 5: Removing values from a set");
            Set<String> tagsToRemove = new HashSet<>();
            tagsToRemove.add("Gadget");

            UpdateItemResponse removeResponse = removeFromSet(
                dynamoDbClient, tableName, key, "Tags", tagsToRemove, false // Not a
number set
                );

            System.out.println("Updated set after removal: " +
removeResponse.attributes());

            // Example 6: Check if a value exists in a set
            System.out.println("\nExample 6: Checking if a value exists in a set");
            Map<String, Object> checkResult = checkIfValueInSet(dynamoDbClient,
tableName, key, "Tags", "Electronics");

            System.out.println("Check result: " + checkResult.get("message"));

            // Example 7: Get the current value of a set attribute
            System.out.println("\nExample 7: Getting the current value of a set
attribute");
            AttributeValue currentStringSet = getSetAttribute(dynamoDbClient,
tableName, key, "Tags");

            if (currentStringSet != null && currentStringSet.ss() != null) {
                System.out.println("Current string set values: " +
currentStringSet.ss());
            } else {
                System.out.println("String set attribute not found");
            }

            AttributeValue currentNumberSet = getSetAttribute(dynamoDbClient,
tableName, key, "Ratings");
```

```
            if (currentNumberSet != null && currentNumberSet.ns() != null) {
                System.out.println("Current number set values: " +
currentNumberSet.ns());
            } else {
                System.out.println("Number set attribute not found");
            }

            // Explain set operations
            System.out.println("\nKey points about DynamoDB set operations:");
            System.out.println(
                "1. DynamoDB supports three set types: string sets (SS), number sets
(NS), and binary sets (BS)");
            System.out.println("2. Sets can only contain elements of the same
type");
            System.out.println("3. Use ADD to add elements to a set");
            System.out.println("4. Use DELETE to remove elements from a set");
            System.out.println("5. Sets automatically remove duplicate values");
            System.out.println("6. Sets are unordered collections");
            System.out.println("7. Use the contains function to check if a value
exists in a set");
            System.out.println("8. You can create a set with multiple values in a
single operation");

        } catch (DynamoDbException e) {
            System.err.println("Error: " + e.getMessage());
            e.printStackTrace();
        }
    }
```

- Per i dettagli sull'API, vedere UpdateItemin AWS SDK for Java 2.x API Reference.

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 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 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);

        // Update multiple movies by using the BatchExecute statement.
        String title1 = "Star Wars";
        int year1 = 1977;
        String title2 = "Wizard of Oz";
        int year2 = 1939;

        System.out.println("Query two movies.");
        getBatch(ddb, tableName, title1, title2, year1, year2);

        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 boolean getBatch(DynamoDbClient ddb, String tableName, String
title1, String title2, int year1, int year2) {
        String getBatch = "SELECT * FROM " + tableName + " WHERE title = ? AND year
= ?";

        List<BatchStatementRequest> statements = new ArrayList<>();
        statements.add(BatchStatementRequest.builder()
            .statement(getBatch)
            .parameters(AttributeValue.builder().s(title1).build(),
                AttributeValue.builder().n(String.valueOf(year1)).build())
            .build());
        statements.add(BatchStatementRequest.builder()
            .statement(getBatch)
            .parameters(AttributeValue.builder().s(title2).build(),
                AttributeValue.builder().n(String.valueOf(year2)).build())
            .build());

        BatchExecuteStatementRequest batchExecuteStatementRequest =
BatchExecuteStatementRequest.builder()
            .statements(statements)
            .build();

        try {
            BatchExecuteStatementResponse response =
ddb.batchExecuteStatement(batchExecuteStatementRequest);
            if (!response.responses().isEmpty()) {
                response.responses().forEach(r -> {
                    System.out.println(r.item().get("title") + "\\t" +
r.item().get("year"));
                });
                return true;
            } else {
                System.out.println("Couldn't find either " + title1 + " or " +
title2 + ".");
                return false;
            }
        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            return false;
        }
    }
```

```
    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)
            .billingMode(BillingMode.PAY_PER_REQUEST) //  DynamoDB automatically
scales based on traffic.
            .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("1977")
                .build();

            AttributeValue att2 = AttributeValue.builder()
                .s("Star Wars")
                .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("1939")
                .build();

            AttributeValue attMovie2A = AttributeValue.builder()
                .s("Wizard of Oz")
                .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, consulta la BatchExecuteStatementsezione 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 {
        String fileName = "../../../resources/sample_files/movies.json";
        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)
            .billingMode(BillingMode.PAY_PER_REQUEST) //Scales based on traffic.
```

```
            .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, consulta la ExecuteStatementsezione AWS SDK for Java 2.x API Reference.

Interroga una tabella utilizzando un indice secondario globale

Il seguente esempio di codice mostra come eseguire una query su una tabella utilizzando un indice secondario globale.

- Interroga una tabella DynamoDB utilizzando la sua chiave primaria.
- Interroga un Global Secondary Index (GSI) per modelli di accesso alternativi.
- Confronta le query relative alle tabelle e le interrogazioni GSI.

SDK per Java 2.x

Interroga una tabella DynamoDB utilizzando la sua chiave primaria e un Global Secondary Index (GSI) con. AWS SDK for Java 2.x

```
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 java.util.HashMap;
import java.util.Map;

    public QueryResponse queryTable(
        final String tableName, final String partitionKeyName, final String
 partitionKeyValue) {

        CodeSampleUtils.validateTableParameters(tableName, partitionKeyName,
 partitionKeyValue);

        // Create expression attribute names for the column names
        final Map<String, String> expressionAttributeNames = new HashMap<>();
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_PK,
 partitionKeyName);

        // Create expression attribute values for the column values
```

```
        final Map<String, AttributeValue> expressionAttributeValues = new
HashMap<>();
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_PK,
            AttributeValue.builder().s(partitionKeyValue).build());

        // Create the query request
        final QueryRequest queryRequest = QueryRequest.builder()
            .tableName(tableName)
            .keyConditionExpression(KEY_CONDITION_EXPRESSION)
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .build();

        try {
            final QueryResponse response = dynamoDbClient.query(queryRequest);
            System.out.println("Query on base table successful. Found " +
response.count() + " items");
            return response;
        } catch (ResourceNotFoundException e) {
            System.err.format("Error: The Amazon DynamoDB table \"%s\" can't be
found.\n", tableName);
            throw new DynamoDbQueryException("Table not found: " + tableName, e);
        } catch (DynamoDbException e) {
            System.err.println("Error querying base table: " + e.getMessage());
            throw new DynamoDbQueryException("Failed to execute query on base
table", e);
        }
    }

    /**
     * Queries a DynamoDB Global Secondary Index (GSI) by partition key.
     *
     * @param tableName         The name of the DynamoDB table
     * @param indexName         The name of the GSI
     * @param partitionKeyName  The name of the GSI partition key attribute
     * @param partitionKeyValue The value of the GSI partition key to query
     * @return The query response from DynamoDB
     * @throws ResourceNotFoundException if the table or index doesn't exist
     * @throws DynamoDbException if the query fails
     */
    public QueryResponse queryGlobalSecondaryIndex(
        final String tableName, final String indexName, final String
partitionKeyName, final String partitionKeyValue) {
```

```
        CodeSampleUtils.validateTableParameters(tableName, partitionKeyName,
partitionKeyValue);
        CodeSampleUtils.validateStringParameter("Index name", indexName);

        // Create expression attribute names for the column names
        final Map<String, String> expressionAttributeNames = new HashMap<>();
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_IK,
partitionKeyName);

        // Create expression attribute values for the column values
        final Map<String, AttributeValue> expressionAttributeValues = new
HashMap<>();
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_IK,
            AttributeValue.builder().s(partitionKeyValue).build());

        // Create the query request
        final QueryRequest queryRequest = QueryRequest.builder()
            .tableName(tableName)
            .indexName(indexName)
            .keyConditionExpression(GSI_KEY_CONDITION_EXPRESSION)
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .build();

        try {
            final QueryResponse response = dynamoDbClient.query(queryRequest);
            System.out.println("Query on GSI successful. Found " + response.count()
+ " items");
            return response;
        } catch (ResourceNotFoundException e) {
            System.err.format(
                "Error: The Amazon DynamoDB table \"%s\" or index \"%s\" can't be
found.\n", tableName, indexName);
            throw new DynamoDbQueryException("Table or index not found: " +
tableName + "/" + indexName, e);
        } catch (DynamoDbException e) {
            System.err.println("Error querying GSI: " + e.getMessage());
            throw new DynamoDbQueryException("Failed to execute query on GSI", e);
        }
    }
```

Confronta l'interrogazione diretta di una tabella con l'interrogazione di un GSI con. AWS SDK for Java 2.x

```
   public static void main(String[] args) {
       final String usage =
           """
               Usage:
                   <tableName> <basePartitionKeyName> <basePartitionKeyValue>
<gsiName> <gsiPartitionKeyName> <gsiPartitionKeyValue> [region]
               Where:
                   tableName - The Amazon DynamoDB table to query.
                   basePartitionKeyName - The name of the base table partition key
attribute.
                   basePartitionKeyValue - The value of the base table partition
key to query.
                   gsiName - The name of the Global Secondary Index.
                   gsiPartitionKeyName - The name of the GSI partition key
attribute.
                   gsiPartitionKeyValue - The value of the GSI partition key to
query.
                   region (optional) - The AWS region where the table exists.
(Default: us-east-1)
               """;

       if (args.length < 6) {
           System.out.println(usage);
           System.exit(1);
       }

       final String tableName = args[0];
       final String basePartitionKeyName = args[1];
       final String basePartitionKeyValue = args[2];
       final String gsiName = args[3];
       final String gsiPartitionKeyName = args[4];
       final String gsiPartitionKeyValue = args[5];
       final Region region = args.length > 6 ? Region.of(args[6]) :
Region.US_EAST_1;

       try (DynamoDbClient ddb = DynamoDbClient.builder().region(region).build()) {
           final QueryTableAndGSI queryHelper = new QueryTableAndGSI(ddb);

           // Query the base table
```

```
            System.out.println("Querying base table where " + basePartitionKeyName +
" = " + basePartitionKeyValue);
            final QueryResponse tableResponse =
                queryHelper.queryTable(tableName, basePartitionKeyName,
basePartitionKeyValue);

            System.out.println("Found " + tableResponse.count() + " items in base
table:");
            tableResponse.items().forEach(item -> System.out.println(item));

            // Query the GSI
            System.out.println(
                "\nQuerying GSI '" + gsiName + "' where " + gsiPartitionKeyName + "
= " + gsiPartitionKeyValue);
            final QueryResponse gsiResponse =
                queryHelper.queryGlobalSecondaryIndex(tableName, gsiName,
gsiPartitionKeyName, gsiPartitionKeyValue);

            System.out.println("Found " + gsiResponse.count() + " items in GSI:");
            gsiResponse.items().forEach(item -> System.out.println(item));

            // Explain the differences between querying a table and a GSI
            System.out.println("\nKey differences between querying a table and a
GSI:");
            System.out.println("1. When querying a GSI, you must specify the
indexName parameter");
            System.out.println("2. GSIs may not contain all attributes from the base
table (projection)");
            System.out.println("3. GSIs consume read capacity units from the GSI's
capacity, not the base table's");
            System.out.println("4. GSIs may have eventually consistent data (cannot
use ConsistentRead=true)");

        } catch (IllegalArgumentException e) {
            System.err.println("Invalid input: " + e.getMessage());
            System.exit(1);
        } catch (ResourceNotFoundException e) {
            System.err.println("Table or index not found: " + e.getMessage());
            System.exit(1);
        } catch (DynamoDbException e) {
            System.err.println("DynamoDB error: " + e.getMessage());
            System.exit(1);
        } catch (Exception e) {
            System.err.println("Unexpected error: " + e.getMessage());
```

```
            System.exit(1);
        }
    }
```

- Per ulteriori informazioni sulle API, consulta Query nella Documentazione di riferimento delle API AWS SDK for Java 2.x .

Interroga una tabella utilizzando una condizione begins_with

Il seguente esempio di codice mostra come interrogare una tabella utilizzando una condizione begins_with.

- Utilizzate la funzione begins_with in un'espressione di condizione chiave.
- Filtra gli elementi in base a uno schema di prefisso nella chiave di ordinamento.

SDK per Java 2.x

Interroga una tabella DynamoDB utilizzando una condizione begins_with sulla chiave di ordinamento con. AWS SDK for Java 2.x

```
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 java.util.HashMap;
import java.util.Map;
import java.util.logging.Level;
import java.util.logging.Logger;

    public QueryResponse queryWithBeginsWithCondition(
        final String tableName,
        final String partitionKeyName,
        final String partitionKeyValue,
        final String sortKeyName,
        final String sortKeyPrefix) {
```

```
        CodeSampleUtils.validateTableParameters(tableName, partitionKeyName,
partitionKeyValue);
        CodeSampleUtils.validateStringParameter("Sort key name", sortKeyName);
        CodeSampleUtils.validateStringParameter("Sort key prefix", sortKeyPrefix);

        // Create expression attribute names for the column names
        final Map<String, String> expressionAttributeNames = new HashMap<>();
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_PK,
partitionKeyName);
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_SK, sortKeyName);

        // Create expression attribute values for the column values
        final Map<String, AttributeValue> expressionAttributeValues = new
HashMap<>();
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_PK,
            AttributeValue.builder().s(partitionKeyValue).build());
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_SK_PREFIX,
            AttributeValue.builder().s(sortKeyPrefix).build());

        // Create the query request
        final QueryRequest queryRequest = QueryRequest.builder()
            .tableName(tableName)
            .keyConditionExpression(KEY_CONDITION_EXPRESSION)
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .build();

        try {
            final QueryResponse response = dynamoDbClient.query(queryRequest);
            LOGGER.log(Level.INFO, "Query with begins_with condition successful.
Found {0} items", response.count());
            return response;
        } catch (ResourceNotFoundException e) {
            LOGGER.log(Level.SEVERE, "Table not found: {0}", tableName);
            throw e;
        } catch (DynamoDbException e) {
            LOGGER.log(Level.SEVERE, "Error querying with begins_with condition",
e);
            throw e;
        }
    }
```

Dimostrate di usare begins_with con prefissi di diversa lunghezza con. AWS SDK for Java 2.x

```
   public static void main(String[] args) {
       try {
           CodeSampleUtils.BeginsWithQueryConfig config =
CodeSampleUtils.BeginsWithQueryConfig.fromArgs(args);
           LOGGER.log(Level.INFO, "Querying items where {0} = {1} and {2} begins
with ''{3}''", new Object[] {
               config.getPartitionKeyName(),
               config.getPartitionKeyValue(),
               config.getSortKeyName(),
               config.getSortKeyPrefix()
           });

           // Using the builder pattern to create and execute the query
           final QueryResponse response = new BeginsWithQueryBuilder()
               .withTableName(config.getTableName())
               .withPartitionKeyName(config.getPartitionKeyName())
               .withPartitionKeyValue(config.getPartitionKeyValue())
               .withSortKeyName(config.getSortKeyName())
               .withSortKeyPrefix(config.getSortKeyPrefix())
               .withRegion(config.getRegion())
               .execute();

           // Process the results
           LOGGER.log(Level.INFO, "Found {0} items:", response.count());
           response.items().forEach(item -> LOGGER.info(item.toString()));

           // Demonstrate with a different prefix
           if (!config.getSortKeyPrefix().isEmpty()) {
               String shorterPrefix = config.getSortKeyPrefix()
                   .substring(0, Math.max(1, config.getSortKeyPrefix().length() /
2));
               LOGGER.log(Level.INFO, "\nNow querying with a shorter prefix:
''{0}''", shorterPrefix);

               final QueryResponse response2 = new BeginsWithQueryBuilder()
                   .withTableName(config.getTableName())
                   .withPartitionKeyName(config.getPartitionKeyName())
                   .withPartitionKeyValue(config.getPartitionKeyValue())
                   .withSortKeyName(config.getSortKeyName())
```

```
                    .withSortKeyPrefix(shorterPrefix)
                    .withRegion(config.getRegion())
                    .execute();

                LOGGER.log(Level.INFO, "Found {0} items with shorter prefix:",
response2.count());
                response2.items().forEach(item -> LOGGER.info(item.toString()));
            }
        } catch (IllegalArgumentException e) {
            LOGGER.log(Level.SEVERE, "Invalid input: {0}", e.getMessage());
            printUsage();
        } catch (ResourceNotFoundException e) {
            LOGGER.log(Level.SEVERE, "Table not found", e);
        } catch (DynamoDbException e) {
            LOGGER.log(Level.SEVERE, "DynamoDB error", e);
        } catch (Exception e) {
            LOGGER.log(Level.SEVERE, "Unexpected error", e);
        }
    }
```

- Per ulteriori informazioni sulle API, consulta Query nella Documentazione di riferimento delle API AWS SDK for Java 2.x .

Interroga una tabella utilizzando un intervallo di date

Il seguente esempio di codice mostra come eseguire una query su una tabella utilizzando un intervallo di date nella chiave di ordinamento.

- Interroga gli elementi all'interno di un intervallo di date specifico.
- Utilizza gli operatori di confronto sulle chiavi di ordinamento in formato data.

SDK per Java 2.x

Interroga una tabella DynamoDB per gli elementi compresi in un intervallo di date con. AWS SDK for Java 2.x

```
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 java.time.LocalDate;
import java.util.HashMap;
import java.util.Map;
import java.util.logging.Level;
import java.util.logging.Logger;

    public QueryResponse queryWithDateRange(
        final String tableName,
        final String partitionKeyName,
        final String partitionKeyValue,
        final String dateKeyName,
        final LocalDate startDate,
        final LocalDate endDate) {

        // Focus on query logic, assuming parameters are valid
        if (startDate == null || endDate == null) {
            throw new IllegalArgumentException("Start date and end date cannot be
 null");
        }

        if (endDate.isBefore(startDate)) {
            throw new IllegalArgumentException("End date must be after start date");
        }

        // Format dates as ISO strings for DynamoDB (using just the date part)
        final String formattedStartDate = startDate.toString();
        final String formattedEndDate = endDate.toString();

        // Create expression attribute names for the column names
        final Map<String, String> expressionAttributeNames = new HashMap<>();
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_PK,
 partitionKeyName);
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_SK, dateKeyName);

        // Create expression attribute values for the column values
        final Map<String, AttributeValue> expressionAttributeValues = new
 HashMap<>();
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_PK,
            AttributeValue.builder().s(partitionKeyValue).build());
```

```
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_START_DATE,
            AttributeValue.builder().s(formattedStartDate).build());
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_END_DATE,
            AttributeValue.builder().s(formattedEndDate).build());

        // Create the query request
        final QueryRequest queryRequest = QueryRequest.builder()
            .tableName(tableName)
            .keyConditionExpression(KEY_CONDITION_EXPRESSION)
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .build();

        try {
            final QueryResponse response = dynamoDbClient.query(queryRequest);
            LOGGER.log(Level.INFO, "Query by date range successful. Found {0}
items", response.count());
            return response;
        } catch (ResourceNotFoundException e) {
            LOGGER.log(Level.SEVERE, "Table not found: {0}", tableName);
            throw e;
        } catch (DynamoDbException e) {
            LOGGER.log(Level.SEVERE, "Error querying by date range: {0}",
e.getMessage());
            throw e;
        }
    }
```

Dimostra come interrogare una tabella DynamoDB con il filtraggio degli intervalli di date.

```
    public static void main(String[] args) {
        final String usage =
            """
                Usage:
                    <tableName> <partitionKeyName> <partitionKeyValue> <dateKeyName>
<startDate> <endDate> [region]
                Where:
                    tableName - The Amazon DynamoDB table to query.
                    partitionKeyName - The name of the partition key attribute.
                    partitionKeyValue - The value of the partition key to query.
```

```
                    dateKeyName - The name of the date attribute to filter on.
                    startDate - The start date for the range query (YYYY-MM-DD).
                    endDate - The end date for the range query (YYYY-MM-DD).
                    region (optional) - The AWS region where the table exists.
(Default: us-east-1)
                """;

        if (args.length < 6) {
            System.out.println(usage);
            System.exit(1);
        }

        try {
            // Parse command line arguments into a config object
            CodeSampleUtils.DateRangeQueryConfig config =
CodeSampleUtils.DateRangeQueryConfig.fromArgs(args);

            LOGGER.log(
                Level.INFO, "Querying items from {0} to {1}", new Object[]
{config.getStartDate(), config.getEndDate()
                });

            // Using the builder pattern to create and execute the query
            final QueryResponse response = new DateRangeQueryBuilder()
                .withTableName(config.getTableName())
                .withPartitionKeyName(config.getPartitionKeyName())
                .withPartitionKeyValue(config.getPartitionKeyValue())
                .withDateKeyName(config.getDateKeyName())
                .withStartDate(config.getStartDate())
                .withEndDate(config.getEndDate())
                .withRegion(config.getRegion())
                .execute();

            // Process the results
            LOGGER.log(Level.INFO, "Found {0} items:", response.count());
            response.items().forEach(item -> {
                LOGGER.info(item.toString());

                // Extract and display the date attribute for clarity
                if (item.containsKey(config.getDateKeyName())) {
                    LOGGER.log(
                        Level.INFO,
                        "  Date attribute: {0}",
                        item.get(config.getDateKeyName()).s());
```

```
                }
            });

            // Demonstrate with a different date range
            LocalDate narrowerStartDate = config.getStartDate().plusDays(1);
            LocalDate narrowerEndDate = config.getEndDate().minusDays(1);

            if (!narrowerStartDate.isAfter(narrowerEndDate)) {
                LOGGER.log(Level.INFO, "\nNow querying with a narrower date range:
{0} to {1}", new Object[] {
                    narrowerStartDate, narrowerEndDate
                });

                final QueryResponse response2 = new DateRangeQueryBuilder()
                    .withTableName(config.getTableName())
                    .withPartitionKeyName(config.getPartitionKeyName())
                    .withPartitionKeyValue(config.getPartitionKeyValue())
                    .withDateKeyName(config.getDateKeyName())
                    .withStartDate(narrowerStartDate)
                    .withEndDate(narrowerEndDate)
                    .withRegion(config.getRegion())
                    .execute();

                LOGGER.log(Level.INFO, "Found {0} items with narrower date range:",
response2.count());
                response2.items().forEach(item -> LOGGER.info(item.toString()));
            }

            LOGGER.info("\nNote: When storing dates in DynamoDB:");
            LOGGER.info("1. Use ISO format (YYYY-MM-DD) for lexicographical
ordering");
            LOGGER.info("2. Use the BETWEEN operator for inclusive date range
queries");
            LOGGER.info("3. Consider using ISO-8601 format for timestamps with time
components");

        } catch (IllegalArgumentException e) {
            LOGGER.log(Level.SEVERE, "Invalid input: {0}", e.getMessage());
            System.exit(1);
        } catch (ResourceNotFoundException e) {
            LOGGER.log(Level.SEVERE, "Table not found: {0}", e.getMessage());
            System.exit(1);
        } catch (DynamoDbException e) {
            LOGGER.log(Level.SEVERE, "DynamoDB error: {0}", e.getMessage());
```

```
            System.exit(1);
        } catch (Exception e) {
            LOGGER.log(Level.SEVERE, "Unexpected error: {0}", e.getMessage());
            System.exit(1);
        }
    }
```

- Per ulteriori informazioni sulle API, consulta Query nella Documentazione di riferimento delle API AWS SDK for Java 2.x .

Interroga una tabella con un'espressione di filtro complessa

Il seguente esempio di codice mostra come eseguire una query su una tabella con un'espressione di filtro complessa.

- Applica espressioni di filtro complesse ai risultati delle query.
- Combina più condizioni utilizzando operatori logici.
- Filtra gli elementi in base ad attributi non chiave.

SDK per Java 2.x

Interroga una tabella DynamoDB con un'espressione di filtro complessa utilizzando. AWS SDK for Java 2.x

```
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 java.util.HashMap;
import java.util.Map;
import java.util.logging.Level;
import java.util.logging.Logger;

    public QueryResponse queryWithComplexFilter(
        final String tableName,
        final String partitionKeyName,
```

```
        final String partitionKeyValue,
        final String statusAttrName,
        final String activeStatus,
        final String pendingStatus,
        final String priceAttrName,
        final double minPrice,
        final double maxPrice,
        final String categoryAttrName) {

        // Validate parameters
        CodeSampleUtils.validateTableParameters(tableName, partitionKeyName,
partitionKeyValue);
        CodeSampleUtils.validateStringParameter("Status attribute name",
statusAttrName);
        CodeSampleUtils.validateStringParameter("Active status", activeStatus);
        CodeSampleUtils.validateStringParameter("Pending status", pendingStatus);
        CodeSampleUtils.validateStringParameter("Price attribute name",
priceAttrName);
        CodeSampleUtils.validateStringParameter("Category attribute name",
categoryAttrName);
        CodeSampleUtils.validateNumericRange("Minimum price", minPrice, 0.0,
Double.MAX_VALUE);
        CodeSampleUtils.validateNumericRange("Maximum price", maxPrice, minPrice,
Double.MAX_VALUE);

        // Create expression attribute names for the column names
        final Map<String, String> expressionAttributeNames = new HashMap<>();
        expressionAttributeNames.put("#pk", partitionKeyName);
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_STATUS,
statusAttrName);
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_PRICE,
priceAttrName);
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_CATEGORY,
categoryAttrName);

        // Create expression attribute values for the column values
        final Map<String, AttributeValue> expressionAttributeValues = new
HashMap<>();
        expressionAttributeValues.put(
            ":pkValue", AttributeValue.builder().s(partitionKeyValue).build());
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_ACTIVE,
            AttributeValue.builder().s(activeStatus).build());
        expressionAttributeValues.put(
```

```
            EXPRESSION_ATTRIBUTE_VALUE_PENDING,
            AttributeValue.builder().s(pendingStatus).build());
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_MIN_PRICE,
            AttributeValue.builder().n(String.valueOf(minPrice)).build());
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_MAX_PRICE,
            AttributeValue.builder().n(String.valueOf(maxPrice)).build());

        // Create the query request
        final QueryRequest queryRequest = QueryRequest.builder()
            .tableName(tableName)
            .keyConditionExpression(KEY_CONDITION_EXPRESSION)
            .filterExpression(FILTER_EXPRESSION)
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .build();

        return dynamoDbClient.query(queryRequest);
    }
```

- Per ulteriori informazioni sulle API, consulta Query nella Documentazione di riferimento delle API AWS SDK for Java 2.x .

Interroga una tabella con un'espressione di filtro dinamica

Il seguente esempio di codice mostra come eseguire una query su una tabella con un'espressione di filtro dinamica.

- Crea espressioni di filtro in modo dinamico in fase di esecuzione.
- Costruisci condizioni di filtro in base all'input dell'utente o allo stato dell'applicazione.
- Aggiungi o rimuovi i criteri di filtro in modo condizionale.

SDK per Java 2.x

Interroga una tabella DynamoDB con un'espressione di filtro costruita dinamicamente utilizzando. AWS SDK for Java 2.x

```
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 java.util.HashMap;
import java.util.Map;

    public static QueryResponse queryWithDynamicFilter(
        final String tableName,
        final String partitionKeyName,
        final String partitionKeyValue,
        final Map<String, Object> filterCriteria,
        final Region region,
        final DynamoDbClient dynamoDbClient) {

        validateParameters(tableName, partitionKeyName, partitionKeyValue,
 filterCriteria);

        DynamoDbClient ddbClient = dynamoDbClient;
        boolean shouldClose = false;

        try {
            if (ddbClient == null) {
                ddbClient = createClient(region);
                shouldClose = true;
            }

            final QueryWithDynamicFilter queryHelper = new
 QueryWithDynamicFilter(ddbClient);
            return queryHelper.queryWithDynamicFilter(tableName, partitionKeyName,
 partitionKeyValue, filterCriteria);
        } catch (ResourceNotFoundException e) {
            System.err.println("Table not found: " + tableName);
            throw e;
        } catch (DynamoDbException e) {
            System.err.println("Failed to execute dynamic filter query: " +
 e.getMessage());
            throw e;
        } catch (Exception e) {
            System.err.println("Unexpected error during query: " + e.getMessage());
            throw e;
        } finally {
```

```
            if (shouldClose && ddbClient != null) {
                ddbClient.close();
            }
        }
    }
```

Dimostra come utilizzare le espressioni di filtro dinamiche con. AWS SDK for Java 2.x

```
    public static void main(String[] args) {
        final String usage =
            """
                Usage:
                    <tableName> <partitionKeyName> <partitionKeyValue>
<filterAttrName> <filterAttrValue> [region]
                Where:
                    tableName - The Amazon DynamoDB table to query.
                    partitionKeyName - The name of the partition key attribute.
                    partitionKeyValue - The value of the partition key to query.
                    filterAttrName - The name of the attribute to filter on.
                    filterAttrValue - The value to filter by.
                    region (optional) - The AWS region where the table exists.
(Default: us-east-1)
                """;

        if (args.length < 5) {
            System.out.println(usage);
            System.exit(1);
        }

        final String tableName = args[0];
        final String partitionKeyName = args[1];
        final String partitionKeyValue = args[2];
        final String filterAttrName = args[3];
        final String filterAttrValue = args[4];
        final Region region = args.length > 5 ? Region.of(args[5]) :
Region.US_EAST_1;

        System.out.println("Querying items with dynamic filter: " + filterAttrName +
" = " + filterAttrValue);

        try {
            // Using the builder pattern to create and execute the query
```

```
            final QueryResponse response = new DynamicFilterQueryBuilder()
                .withTableName(tableName)
                .withPartitionKeyName(partitionKeyName)
                .withPartitionKeyValue(partitionKeyValue)
                .withFilterCriterion(filterAttrName, filterAttrValue)
                .withRegion(region)
                .execute();

            // Process the results
            System.out.println("Found " + response.count() + " items:");
            response.items().forEach(item -> System.out.println(item));

            // Demonstrate multiple filter criteria
            System.out.println("\nNow querying with multiple filter criteria:");

            Map<String, Object> multipleFilters = new HashMap<>();
            multipleFilters.put(filterAttrName, filterAttrValue);
            multipleFilters.put("status", "active");

            final QueryResponse multiFilterResponse = new
DynamicFilterQueryBuilder()
                .withTableName(tableName)
                .withPartitionKeyName(partitionKeyName)
                .withPartitionKeyValue(partitionKeyValue)
                .withFilterCriteria(multipleFilters)
                .withRegion(region)
                .execute();

            System.out.println("Found " + multiFilterResponse.count() + " items with
multiple filters:");
            multiFilterResponse.items().forEach(item -> System.out.println(item));

        } catch (IllegalArgumentException e) {
            System.err.println("Invalid input: " + e.getMessage());
            System.exit(1);
        } catch (ResourceNotFoundException e) {
            System.err.println("Table not found: " + tableName);
            System.exit(1);
        } catch (DynamoDbException e) {
            System.err.println("DynamoDB error: " + e.getMessage());
            System.exit(1);
        } catch (Exception e) {
            System.err.println("Unexpected error: " + e.getMessage());
            System.exit(1);
```

```
        }
    }
```

- Per ulteriori informazioni sulle API, consulta Query nella Documentazione di riferimento delle API AWS SDK for Java 2.x .

Interroga una tabella con un'espressione di filtro e un limite

Il seguente esempio di codice mostra come eseguire una query su una tabella con un'espressione di filtro e un limite.

- Applica le espressioni di filtro ai risultati delle query con un limite sugli elementi valutati.
- Scopri in che modo il limite influisce sui risultati delle query filtrate.
- Controlla il numero massimo di elementi elaborati in una query.

SDK per Java 2.x

Interroga una tabella DynamoDB con un'espressione di filtro e limita l'utilizzo. AWS SDK for Java 2.x

```
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 java.util.HashMap;
import java.util.Map;
import java.util.logging.Level;
import java.util.logging.Logger;

    public QueryResponse queryWithFilterAndLimit(
        final String tableName,
        final String partitionKeyName,
        final String partitionKeyValue,
        final String filterAttrName,
        final String filterAttrValue,
        final int limit) {
```

```
        CodeSampleUtils.validateTableParameters(tableName, partitionKeyName,
partitionKeyValue);
        CodeSampleUtils.validateStringParameter("Filter attribute name",
filterAttrName);
        CodeSampleUtils.validateStringParameter("Filter attribute value",
filterAttrValue);
        CodeSampleUtils.validatePositiveInteger("Limit", limit);

        // Create expression attribute names for the column names
        final Map<String, String> expressionAttributeNames = new HashMap<>();
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_PK,
partitionKeyName);
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_FILTER,
filterAttrName);

        // Create expression attribute values for the column values
        final Map<String, AttributeValue> expressionAttributeValues = new
HashMap<>();
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_PK,
            AttributeValue.builder().s(partitionKeyValue).build());
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_FILTER,
            AttributeValue.builder().s(filterAttrValue).build());

        // Create the filter expression
        final String filterExpression = "#filterAttr = :filterValue";

        // Create the query request
        final QueryRequest queryRequest = QueryRequest.builder()
            .tableName(tableName)
            .keyConditionExpression(KEY_CONDITION_EXPRESSION)
            .filterExpression(filterExpression)
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .limit(limit)
            .build();

        try {
            final QueryResponse response = dynamoDbClient.query(queryRequest);
            LOGGER.log(Level.INFO, "Query with filter and limit successful. Found
{0} items", response.count());
            LOGGER.log(
```

```
                Level.INFO, "ScannedCount: {0} (total items evaluated before
filtering)", response.scannedCount());
            return response;
        } catch (ResourceNotFoundException e) {
            LOGGER.log(Level.SEVERE, "Table not found: {0}", tableName);
            throw e;
        } catch (DynamoDbException e) {
            LOGGER.log(Level.SEVERE, "Error querying with filter and limit: {0}",
e.getMessage());
            throw e;
        }
    }
```

- Per ulteriori informazioni sulle API, consulta Query nella Documentazione di riferimento delle API AWS SDK for Java 2.x .

Interroga una tabella con attributi annidati

Il seguente esempio di codice mostra come interrogare una tabella con attributi annidati.

- Accedi e filtra in base agli attributi annidati negli elementi DynamoDB.
- Utilizza le espressioni del percorso del documento per fare riferimento agli elementi nidificati.

SDK per Java 2.x

Interroga una tabella DynamoDB con attributi annidati utilizzando. AWS SDK for Java 2.x

```
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 java.util.HashMap;
import java.util.Map;

    public QueryResponse queryWithNestedAttributes(
        final String tableName,
        final String partitionKeyName,
```

```
        final String partitionKeyValue,
        final String nestedPath,
        final String nestedAttr,
        final String nestedValue) {

        CodeSampleUtils.validateTableParameters(tableName, partitionKeyName,
partitionKeyValue);
        CodeSampleUtils.validateStringParameter("Nested path", nestedPath);
        CodeSampleUtils.validateStringParameter("Nested attribute", nestedAttr);
        CodeSampleUtils.validateStringParameter("Nested value", nestedValue);

        // Split the nested path into components
        final String[] pathComponents = nestedPath.split("\\.");

        // Create expression attribute names for the column names
        final Map<String, String> expressionAttributeNames = new HashMap<>();
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_PK,
partitionKeyName);

        // Build the nested attribute reference using document path notation
        final StringBuilder nestedAttributeRef = new StringBuilder();
        for (int i = 0; i < pathComponents.length; i++) {
            final String aliasName = "#n" + i;
            expressionAttributeNames.put(aliasName, pathComponents[i]);

            if (i > 0) {
                nestedAttributeRef.append(".");
            }
            nestedAttributeRef.append(aliasName);
        }

        // Create expression attribute values for the column values
        final Map<String, AttributeValue> expressionAttributeValues = new
HashMap<>();
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_PK,
            AttributeValue.builder().s(partitionKeyValue).build());
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_NESTED,
            AttributeValue.builder().s(nestedValue).build());

        // Create the filter expression using the nested attribute reference
        final String filterExpression = nestedAttributeRef + " = :nestedValue";
```

```
        // Create the query request
        final QueryRequest queryRequest = QueryRequest.builder()
            .tableName(tableName)
            .keyConditionExpression(KEY_CONDITION_EXPRESSION)
            .filterExpression(filterExpression)
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .build();

        try {
            final QueryResponse response = dynamoDbClient.query(queryRequest);
            System.out.println("Query with nested attribute filter successful. Found
" + response.count() + " items");
            return response;
        } catch (ResourceNotFoundException e) {
            System.err.format("Error: The Amazon DynamoDB table \"%s\" can't be
found.\n", tableName);
            throw e;
        } catch (DynamoDbException e) {
            System.err.println("Error querying with nested attribute filter: " +
e.getMessage());
            throw e;
        }
    }
```

Dimostra come interrogare una tabella DynamoDB con attributi annidati.

```
    public static void main(String[] args) {
        final String usage =
            """
                Usage:
                    <tableName> <partitionKeyName> <partitionKeyValue> <nestedPath>
<nestedAttr> <nestedValue> [region]
                Where:
                    tableName - The Amazon DynamoDB table to query.
                    partitionKeyName - The name of the partition key attribute.
                    partitionKeyValue - The value of the partition key to query.
                    nestedPath - The path to the nested map attribute (e.g.,
"address").
                    nestedAttr - The name of the nested attribute (e.g., "city").
                    nestedValue - The value to filter by (e.g., "Seattle").
```

```
                    region (optional) - The AWS region where the table exists.
(Default: us-east-1)
                """;

        if (args.length < 6) {
            System.out.println(usage);
            System.exit(1);
        }

        final String tableName = args[0];
        final String partitionKeyName = args[1];
        final String partitionKeyValue = args[2];
        final String nestedPath = args[3];
        final String nestedAttr = args[4];
        final String nestedValue = args[5];
        final Region region = args.length > 6 ? Region.of(args[6]) :
Region.US_EAST_1;

        System.out.println("Querying items where " + partitionKeyName + " = " +
partitionKeyValue + " and " + nestedPath
            + "." + nestedAttr + " = " + nestedValue);

        try {
            // Using the builder pattern to create and execute the query
            final QueryResponse response = new NestedAttributeQueryBuilder()
                .withTableName(tableName)
                .withPartitionKeyName(partitionKeyName)
                .withPartitionKeyValue(partitionKeyValue)
                .withNestedPath(nestedPath)
                .withNestedAttribute(nestedAttr)
                .withNestedValue(nestedValue)
                .withRegion(region)
                .execute();

            // Process the results
            System.out.println("Found " + response.count() + " items:");
            response.items().forEach(item -> {
                System.out.println(item);

                // Extract and display the nested attribute for clarity
                if (item.containsKey(nestedPath) && item.get(nestedPath).hasM()) {
                    Map<String, AttributeValue> nestedMap =
item.get(nestedPath).m();
                    if (nestedMap.containsKey(nestedAttr)) {
```

```
                        System.out.println("  Nested attribute " + nestedPath + "."
+ nestedAttr + ": "
                                + formatAttributeValue(nestedMap.get(nestedAttr)));
                    }
                }
            });

            System.out.println("\nNote: When working with nested attributes in
DynamoDB:");
            System.out.println("1. Use dot notation in filter expressions to access
nested attributes");
            System.out.println("2. Use expression attribute names for each component
of the path");
            System.out.println("3. Check if the nested attribute exists before
accessing it");

        } catch (IllegalArgumentException e) {
            System.err.println("Invalid input: " + e.getMessage());
            System.exit(1);
        } catch (ResourceNotFoundException e) {
            System.err.println("Table not found: " + tableName);
            System.exit(1);
        } catch (DynamoDbException e) {
            System.err.println("DynamoDB error: " + e.getMessage());
            System.exit(1);
        } catch (Exception e) {
            System.err.println("Unexpected error: " + e.getMessage());
            System.exit(1);
        }
    }
```

- Per ulteriori informazioni sulle API, consulta Query nella Documentazione di riferimento delle API AWS SDK for Java 2.x .

Interroga una tabella con impaginazione

Il seguente esempio di codice mostra come interrogare una tabella con impaginazione.

- Implementa la paginazione per i risultati delle query DynamoDB.
- Utilizzate il LastEvaluatedKey per recuperare le pagine successive.
- Controlla il numero di elementi per pagina con il parametro Limit.

SDK per Java 2.x

Interroga una tabella DynamoDB con impaginazione utilizzando. AWS SDK for Java 2.x

```
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 java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

    public List<Map<String, AttributeValue>> queryWithPagination(
        final String tableName, final String partitionKeyName, final String
 partitionKeyValue, final int pageSize) {

        CodeSampleUtils.validateTableParameters(tableName, partitionKeyName,
 partitionKeyValue);
        CodeSampleUtils.validatePositiveInteger("Page size", pageSize);

        // Create expression attribute names for the column names
        final Map<String, String> expressionAttributeNames = new HashMap<>();
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_PK,
 partitionKeyName);

        // Create expression attribute values for the column values
        final Map<String, AttributeValue> expressionAttributeValues = new
 HashMap<>();
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_PK,
            AttributeValue.builder().s(partitionKeyValue).build());

        // Create the query request
        QueryRequest.Builder queryRequestBuilder = QueryRequest.builder()
            .tableName(tableName)
            .keyConditionExpression(KEY_CONDITION_EXPRESSION)
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .limit(pageSize);
```

```
        // List to store all items from all pages
        final List<Map<String, AttributeValue>> allItems = new ArrayList<>();

        // Map to store the last evaluated key for pagination
        Map<String, AttributeValue> lastEvaluatedKey = null;
        int pageNumber = 1;

        try {
            do {
                // If we have a last evaluated key, use it for the next page
                if (lastEvaluatedKey != null) {
                    queryRequestBuilder.exclusiveStartKey(lastEvaluatedKey);
                }

                // Execute the query
                final QueryResponse response =
dynamoDbClient.query(queryRequestBuilder.build());

                // Process the current page of results
                final List<Map<String, AttributeValue>> pageItems =
response.items();
                allItems.addAll(pageItems);

                // Get the last evaluated key for the next page
                lastEvaluatedKey = response.lastEvaluatedKey();
                if (lastEvaluatedKey != null && lastEvaluatedKey.isEmpty()) {
                    lastEvaluatedKey = null;
                }

                System.out.println("Page " + pageNumber + ": Retrieved " +
pageItems.size() + " items (Running total: "
                    + allItems.size() + ")");

                pageNumber++;

            } while (lastEvaluatedKey != null);

            System.out.println("Query with pagination complete. Retrieved a total of
" + allItems.size()
                + " items across " + (pageNumber - 1) + " pages");

            return allItems;
        } catch (ResourceNotFoundException e) {
```

```
            System.err.format("Error: The Amazon DynamoDB table \"%s\" can't be
found.\n", tableName);
            throw e;
        } catch (DynamoDbException e) {
            System.err.println("Error querying with pagination: " + e.getMessage());
            throw e;
        }
    }
```

Dimostra come interrogare una tabella DynamoDB con impaginazione.

```
    public static void main(String[] args) {
        final String usage =
            """
                Usage:
                    <tableName> <partitionKeyName> <partitionKeyValue> [pageSize]
[region]
                Where:
                    tableName - The Amazon DynamoDB table to query.
                    partitionKeyName - The name of the partition key attribute.
                    partitionKeyValue - The value of the partition key to query.
                    pageSize (optional) - The maximum number of items to return per
page. (Default: 10)
                    region (optional) - The AWS region where the table exists.
(Default: us-east-1)
                """;

        if (args.length < 3) {
            System.out.println(usage);
            System.exit(1);
        }

        final String tableName = args[0];
        final String partitionKeyName = args[1];
        final String partitionKeyValue = args[2];
        final int pageSize = args.length > 3 ? Integer.parseInt(args[3]) : 10;
        final Region region = args.length > 4 ? Region.of(args[4]) :
Region.US_EAST_1;

        System.out.println("Querying items with pagination (page size: " + pageSize
+ ")");
```

```
        try {
            // Using the builder pattern to create and execute the query
            final List<Map<String, AttributeValue>> allItems = new
PaginationQueryBuilder()
                .withTableName(tableName)
                .withPartitionKeyName(partitionKeyName)
                .withPartitionKeyValue(partitionKeyValue)
                .withPageSize(pageSize)
                .withRegion(region)
                .executeWithPagination();

            // Process the results
            System.out.println("\nSummary: Retrieved a total of " + allItems.size()
+ " items");

            // Display the first few items as a sample
            final int sampleSize = Math.min(5, allItems.size());
            if (sampleSize > 0) {
                System.out.println("\nSample of retrieved items (first " +
sampleSize + "):");
                for (int i = 0; i < sampleSize; i++) {
                    System.out.println(allItems.get(i));
                }

                if (allItems.size() > sampleSize) {
                    System.out.println("... and " + (allItems.size() - sampleSize) +
" more items");
                }
            }
        } catch (IllegalArgumentException e) {
            System.err.println("Invalid input: " + e.getMessage());
            System.exit(1);
        } catch (ResourceNotFoundException e) {
            System.err.println("Table not found: " + tableName);
            System.exit(1);
        } catch (DynamoDbException e) {
            System.err.println("DynamoDB error: " + e.getMessage());
            System.exit(1);
        } catch (Exception e) {
            System.err.println("Unexpected error: " + e.getMessage());
            System.exit(1);
        }
    }
```

- Per ulteriori informazioni sulle API, consulta Query nella Documentazione di riferimento delle API AWS SDK for Java 2.x .

Interroga una tabella con letture fortemente coerenti

Il seguente esempio di codice mostra come interrogare una tabella con letture fortemente coerenti.

- Configura il livello di coerenza per le query DynamoDB.
- Utilizza letture fortemente coerenti per ottenere la maggior parte dei dati. up-to-date
- Comprendi i compromessi tra coerenza finale e forte coerenza.

SDK per Java 2.x

Interroga una tabella DynamoDB con coerenza di lettura configurabile utilizzando. AWS SDK for Java 2.x

```
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 java.util.HashMap;
import java.util.Map;
import java.util.logging.Level;
import java.util.logging.Logger;

    public QueryResponse queryWithConsistentReads(
        final String tableName,
        final String partitionKeyName,
        final String partitionKeyValue,
        final boolean useConsistentRead) {

        CodeSampleUtils.validateTableParameters(tableName, partitionKeyName,
 partitionKeyValue);

        // Create expression attribute names for the column names
```

```
        final Map<String, String> expressionAttributeNames = new HashMap<>();
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_PK,
partitionKeyName);

        // Create expression attribute values for the column values
        final Map<String, AttributeValue> expressionAttributeValues = new
HashMap<>();
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_PK,
            AttributeValue.builder().s(partitionKeyValue).build());

        // Create the query request
        final QueryRequest queryRequest = QueryRequest.builder()
            .tableName(tableName)
            .keyConditionExpression(KEY_CONDITION_EXPRESSION)
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .consistentRead(useConsistentRead)
            .build();

        try {
            final QueryResponse response = dynamoDbClient.query(queryRequest);
            LOGGER.log(Level.INFO, "Query successful. Found {0} items",
response.count());
            return response;
        } catch (ResourceNotFoundException e) {
            LOGGER.log(Level.SEVERE, "Table not found: {0}", tableName);
            throw e;
        } catch (DynamoDbException e) {
            LOGGER.log(Level.SEVERE, "Error querying with consistent reads", e);
            throw e;
        }
    }
```

- Per ulteriori informazioni sulle API, consulta Query nella Documentazione di riferimento delle API AWS SDK for Java 2.x .

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 utilizzando.
AWS SDK for Java 2.x

```
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 java.util.Map;
import java.util.Optional;

        final QueryRequest request = QueryRequest.builder()
            .tableName(tableName)
            .keyConditionExpression(KEY_CONDITION_EXPRESSION)
            .filterExpression(FILTER_EXPRESSION)
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .build();

        try (DynamoDbClient ddb = dynamoDbClient != null
            ? dynamoDbClient
            : DynamoDbClient.builder().region(region).build()) {
            final QueryResponse response = ddb.query(request);
            System.out.println("Query successful. Found " + response.count() + "
 items that have not expired yet.");

            // Print each item
            response.items().forEach(item -> {
                System.out.println("Item: " + item);
            });

            return 0;
        } catch (ResourceNotFoundException e) {
            System.err.format(TABLE_NOT_FOUND_ERROR, tableName);
            throw e;
        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            throw e;
        }
```

- Per ulteriori informazioni sulle API, consulta Query nella Documentazione di riferimento delle API AWS SDK for Java 2.x .

Interroga le tabelle utilizzando modelli di data e ora

Il seguente esempio di codice mostra come interrogare le tabelle utilizzando modelli di data e ora.

- Archivia e interroga i valori di data/ora in DynamoDB.
- Implementa interrogazioni su intervalli di date utilizzando chiavi di ordinamento.
- Formatta le stringhe di data per un'interrogazione efficace.

SDK per Java 2.x

Esegui una query utilizzando intervalli di date nelle chiavi di ordinamento con. AWS SDK for Java 2.x

```
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 java.time.LocalDate;
import java.util.HashMap;
import java.util.Map;
import java.util.logging.Level;
import java.util.logging.Logger;

    public QueryResponse queryWithDateRange(
        final String tableName,
        final String partitionKeyName,
        final String partitionKeyValue,
        final String dateKeyName,
        final LocalDate startDate,
        final LocalDate endDate) {

        // Focus on query logic, assuming parameters are valid
```

```
        if (startDate == null || endDate == null) {
            throw new IllegalArgumentException("Start date and end date cannot be
null");
        }

        if (endDate.isBefore(startDate)) {
            throw new IllegalArgumentException("End date must be after start date");
        }

        // Format dates as ISO strings for DynamoDB (using just the date part)
        final String formattedStartDate = startDate.toString();
        final String formattedEndDate = endDate.toString();

        // Create expression attribute names for the column names
        final Map<String, String> expressionAttributeNames = new HashMap<>();
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_PK,
partitionKeyName);
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_SK, dateKeyName);

        // Create expression attribute values for the column values
        final Map<String, AttributeValue> expressionAttributeValues = new
HashMap<>();
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_PK,
            AttributeValue.builder().s(partitionKeyValue).build());
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_START_DATE,
            AttributeValue.builder().s(formattedStartDate).build());
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_END_DATE,
            AttributeValue.builder().s(formattedEndDate).build());

        // Create the query request
        final QueryRequest queryRequest = QueryRequest.builder()
            .tableName(tableName)
            .keyConditionExpression(KEY_CONDITION_EXPRESSION)
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .build();

        try {
            final QueryResponse response = dynamoDbClient.query(queryRequest);
            LOGGER.log(Level.INFO, "Query by date range successful. Found {0}
items", response.count());
```

```
            return response;
        } catch (ResourceNotFoundException e) {
            LOGGER.log(Level.SEVERE, "Table not found: {0}", tableName);
            throw e;
        } catch (DynamoDbException e) {
            LOGGER.log(Level.SEVERE, "Error querying by date range: {0}",
 e.getMessage());
            throw e;
        }
    }
```

Interroga utilizzando variabili data-ora con. AWS SDK for Java 2.x

```
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 java.time.Instant;
import java.time.LocalDateTime;
import java.time.ZoneOffset;
import java.util.HashMap;
import java.util.Map;

    public QueryResponse queryWithDateTime(
        final String tableName,
        final String partitionKeyName,
        final String partitionKeyValue,
        final String dateKeyName,
        final String startDate,
        final String endDate) {

        CodeSampleUtils.validateTableParameters(tableName, partitionKeyName,
 partitionKeyValue);
        CodeSampleUtils.validateDateRangeParameters(dateKeyName, startDate,
 endDate);
        CodeSampleUtils.validateDateFormat("Start date", startDate);
        CodeSampleUtils.validateDateFormat("End date", endDate);

        // Create expression attribute names for the column names
```

```
        final Map<String, String> expressionAttributeNames = new HashMap<>();
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_PK,
partitionKeyName);
        expressionAttributeNames.put("#dateKey", dateKeyName);

        // Create expression attribute values for the column values
        final Map<String, AttributeValue> expressionAttributeValues = new
HashMap<>();
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_PK,
            AttributeValue.builder().s(partitionKeyValue).build());
        expressionAttributeValues.put(
            ":startDate", AttributeValue.builder().s(startDate).build());
        expressionAttributeValues.put(
            ":endDate", AttributeValue.builder().s(endDate).build());

        // Create the query request
        final QueryRequest queryRequest = QueryRequest.builder()
            .tableName(tableName)
            .keyConditionExpression(KEY_CONDITION_EXPRESSION)
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .build();

        try {
            final QueryResponse response = dynamoDbClient.query(queryRequest);
            System.out.println("Query successful. Found " + response.count() + "
items");
            return response;
        } catch (ResourceNotFoundException e) {
            System.err.format("Error: The Amazon DynamoDB table \"%s\" can't be
found.\n", tableName);
            throw e;
        } catch (DynamoDbException e) {
            System.err.println("Error querying with date range: " + e.getMessage());
            throw e;
        }
    }
```

Interroga entro intervalli di date nei timestamp di epoca Unix con. AWS SDK for Java 2.x

```
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 java.time.Instant;
import java.time.LocalDateTime;
import java.time.ZoneOffset;
import java.util.HashMap;
import java.util.Map;

    public QueryResponse queryWithDateTimeEpoch(
        final String tableName,
        final String partitionKeyName,
        final String partitionKeyValue,
        final String dateKeyName,
        final long startEpoch,
        final long endEpoch) {

        CodeSampleUtils.validateTableParameters(tableName, partitionKeyName,
 partitionKeyValue);
        CodeSampleUtils.validateStringParameter("Date key name", dateKeyName);
        CodeSampleUtils.validateEpochTimestamp("Start epoch", startEpoch);
        CodeSampleUtils.validateEpochTimestamp("End epoch", endEpoch);

        // Create expression attribute names for the column names
        final Map<String, String> expressionAttributeNames = new HashMap<>();
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_PK,
 partitionKeyName);
        expressionAttributeNames.put("#dateKey", dateKeyName);

        // Create expression attribute values for the column values
        final Map<String, AttributeValue> expressionAttributeValues = new
 HashMap<>();
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_PK,
            AttributeValue.builder().s(partitionKeyValue).build());
        expressionAttributeValues.put(
            ":startDate",
 AttributeValue.builder().n(String.valueOf(startEpoch)).build());
        expressionAttributeValues.put(
            ":endDate",
 AttributeValue.builder().n(String.valueOf(endEpoch)).build());
```

```
        // Create the query request
        final QueryRequest queryRequest = QueryRequest.builder()
            .tableName(tableName)
            .keyConditionExpression(KEY_CONDITION_EXPRESSION)
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .build();

        try {
            final QueryResponse response = dynamoDbClient.query(queryRequest);
            System.out.println("Query successful. Found " + response.count() + "
 items");
            return response;
        } catch (ResourceNotFoundException e) {
            System.err.format("Error: The Amazon DynamoDB table \"%s\" can't be
 found.\n", tableName);
            throw e;
        } catch (DynamoDbException e) {
            System.err.println("Error querying with epoch timestamps: " +
 e.getMessage());
            throw e;
        }
    }
```

Esegui una query entro intervalli di date utilizzando oggetti con. LocalDateTime AWS SDK for Java 2.x

```
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 java.time.Instant;
import java.time.LocalDateTime;
import java.time.ZoneOffset;
import java.util.HashMap;
import java.util.Map;

    public QueryResponse queryWithDateTimeLocalDateTime(
```

```
        final String tableName,
        final String partitionKeyName,
        final String partitionKeyValue,
        final String dateKeyName,
        final LocalDateTime startDateTime,
        final LocalDateTime endDateTime) {

        CodeSampleUtils.validateTableParameters(tableName, partitionKeyName,
partitionKeyValue);
        CodeSampleUtils.validateStringParameter("Date key name", dateKeyName);
        if (startDateTime == null || endDateTime == null) {
            throw new IllegalArgumentException("Start and end LocalDateTime must not
be null");
        }

        // Convert LocalDateTime to ISO-8601 strings in UTC with the correct format
        final String startDate =
startDateTime.atZone(ZoneOffset.UTC).format(DATE_TIME_FORMATTER);
        final String endDate =
endDateTime.atZone(ZoneOffset.UTC).format(DATE_TIME_FORMATTER);

        return queryWithDateTime(tableName, partitionKeyName, partitionKeyValue,
dateKeyName, startDate, endDate);
    }
```

- Per ulteriori informazioni sulle API, consulta Query nella Documentazione di riferimento delle API AWS SDK for Java 2.x .

Comprendi l'ordine di aggiornamento delle espressioni

Il seguente esempio di codice mostra come comprendere l'ordine delle espressioni di aggiornamento.

- Scopri come DynamoDB elabora le espressioni di aggiornamento.
- Comprendi l'ordine delle operazioni nelle espressioni di aggiornamento.
- Evita risultati imprevisti comprendendo la valutazione delle espressioni.

SDK per Java 2.x

Dimostra di aggiornare l'ordine delle espressioni utilizzando AWS SDK for Java 2.x.

```
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.GetItemRequest;
import software.amazon.awssdk.services.dynamodb.model.GetItemResponse;
import software.amazon.awssdk.services.dynamodb.model.ReturnValue;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemRequest;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemResponse;

import java.util.HashMap;
import java.util.Map;

    /**
     * Demonstrates the effect of update expression order.
     *
     * <p>This method shows how the order of operations in an update expression
     * affects the result of the update.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @return Map containing the results of different update orders
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static Map<String, Object> demonstrateUpdateOrder(
        DynamoDbClient dynamoDbClient, String tableName, Map<String, AttributeValue>
 key) {

        Map<String, Object> results = new HashMap<>();

        try {
            // Initialize the item with a counter
            UpdateItemRequest initRequest = UpdateItemRequest.builder()
                .tableName(tableName)
                .key(key)
                .updateExpression("SET Counter = :zero, OldCounter = :zero")
                .expressionAttributeValues(
                    Map.of(":zero", AttributeValue.builder().n("0").build()))
                .returnValues(ReturnValue.UPDATED_NEW)
                .build();

            dynamoDbClient.updateItem(initRequest);
```

```
            // Example 1: SET first, then ADD
            UpdateItemRequest setFirstRequest = UpdateItemRequest.builder()
                .tableName(tableName)
                .key(key)
                .updateExpression("SET Counter = :value ADD OldCounter :increment")
                .expressionAttributeValues(Map.of(
                    ":value", AttributeValue.builder().n("10").build(),
                    ":increment", AttributeValue.builder().n("5").build()))
                .returnValues(ReturnValue.UPDATED_NEW)
                .build();

            UpdateItemResponse setFirstResponse =
dynamoDbClient.updateItem(setFirstRequest);
            results.put("setFirstResponse", setFirstResponse);

            // Reset the item
            dynamoDbClient.updateItem(initRequest);

            // Example 2: ADD first, then SET
            UpdateItemRequest addFirstRequest = UpdateItemRequest.builder()
                .tableName(tableName)
                .key(key)
                .updateExpression("ADD Counter :increment SET OldCounter = :value")
                .expressionAttributeValues(Map.of(
                    ":value", AttributeValue.builder().n("10").build(),
                    ":increment", AttributeValue.builder().n("5").build()))
                .returnValues(ReturnValue.UPDATED_NEW)
                .build();

            UpdateItemResponse addFirstResponse =
dynamoDbClient.updateItem(addFirstRequest);
            results.put("addFirstResponse", addFirstResponse);

            // Reset the item
            dynamoDbClient.updateItem(initRequest);

            // Example 3: SET with multiple attributes
            UpdateItemRequest multiSetRequest = UpdateItemRequest.builder()
                .tableName(tableName)
                .key(key)
                .updateExpression("SET Counter = :value, OldCounter = Counter")
                .expressionAttributeValues(
                    Map.of(":value", AttributeValue.builder().n("10").build()))
                .returnValues(ReturnValue.UPDATED_NEW)
```

```
                .build();

            UpdateItemResponse multiSetResponse =
dynamoDbClient.updateItem(multiSetRequest);
            results.put("multiSetResponse", multiSetResponse);

            // Reset the item
            dynamoDbClient.updateItem(initRequest);

            // Example 4: SET with expression using the same attribute
            UpdateItemRequest selfReferenceRequest = UpdateItemRequest.builder()
                .tableName(tableName)
                .key(key)
                .updateExpression("SET Counter = Counter + :increment, OldCounter =
Counter")
                .expressionAttributeValues(
                    Map.of(":increment", AttributeValue.builder().n("5").build()))
                .returnValues(ReturnValue.UPDATED_NEW)
                .build();

            UpdateItemResponse selfReferenceResponse =
dynamoDbClient.updateItem(selfReferenceRequest);
            results.put("selfReferenceResponse", selfReferenceResponse);

            results.put("success", true);

        } catch (DynamoDbException e) {
            results.put("success", false);
            results.put("error", e.getMessage());
        }

        return results;
    }

    /**
     * Updates an item with SET first, then REMOVE.
     *
     * <p>This method demonstrates updating an item with SET operation first,
     * followed by a REMOVE operation.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param attributeToSet The attribute to set
```

```
     * @param setValue The value to set
     * @param attributeToRemove The attribute to remove
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static UpdateItemResponse updateWithSetFirst(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        String attributeToSet,
        AttributeValue setValue,
        String attributeToRemove) {

        // Define the update parameters
        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .updateExpression("SET #setAttr = :setValue REMOVE #removeAttr")
            .expressionAttributeNames(Map.of(
                "#setAttr", attributeToSet,
                "#removeAttr", attributeToRemove))
            .expressionAttributeValues(Map.of(":setValue", setValue))
            .returnValues(ReturnValue.UPDATED_NEW)
            .build();

        // Perform the update operation
        try {
            return dynamoDbClient.updateItem(request);
        } catch (DynamoDbException e) {
            throw DynamoDbException.builder()
                .message("Failed to update item with SET first: " + e.getMessage())
                .cause(e)
                .build();
        }
    }

    /**
     * Updates an item with REMOVE first, then SET.
     *
     * <p>This method demonstrates updating an item with REMOVE operation first,
     * followed by a SET operation.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
```

```
     * @param key The key of the item to update
     * @param attributeToSet The attribute to set
     * @param setValue The value to set
     * @param attributeToRemove The attribute to remove
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static UpdateItemResponse updateWithRemoveFirst(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        String attributeToSet,
        AttributeValue setValue,
        String attributeToRemove) {

        // Define the update parameters
        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .updateExpression("REMOVE #removeAttr SET #setAttr = :setValue")
            .expressionAttributeNames(Map.of(
                "#setAttr", attributeToSet,
                "#removeAttr", attributeToRemove))
            .expressionAttributeValues(Map.of(":setValue", setValue))
            .returnValues(ReturnValue.UPDATED_NEW)
            .build();

        // Perform the update operation
        try {
            return dynamoDbClient.updateItem(request);
        } catch (DynamoDbException e) {
            throw DynamoDbException.builder()
                .message("Failed to update item with REMOVE first: " +
e.getMessage())
                .cause(e)
                .build();
        }
    }

    /**
     * Updates an item with all operation types in a specific order.
     *
     * <p>This method demonstrates using all operation types (SET, REMOVE, ADD,
DELETE)
```

```
     * in a specific order in a single update expression.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static UpdateItemResponse updateWithAllOperationTypes(
        DynamoDbClient dynamoDbClient, String tableName, Map<String, AttributeValue>
key) {

        // Define the update parameters
        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .updateExpression("SET #stringAttr = :stringVal, #mapAttr.#nestedAttr
= :nestedVal " + "REMOVE #oldAttr "
                + "ADD #counterAttr :increment "
                + "DELETE #stringSetAttr :stringSetVal")
            .expressionAttributeNames(Map.of(
                "#stringAttr", "StringAttribute",
                "#mapAttr", "MapAttribute",
                "#nestedAttr", "NestedAttribute",
                "#oldAttr", "OldAttribute",
                "#counterAttr", "CounterAttribute",
                "#stringSetAttr", "StringSetAttribute"))
            .expressionAttributeValues(Map.of(
                ":stringVal", AttributeValue.builder().s("New Value").build(),
                ":nestedVal", AttributeValue.builder().s("Nested Value").build(),
                ":increment", AttributeValue.builder().n("1").build(),
                ":stringSetVal", AttributeValue.builder().ss("Value1").build()))
            .returnValues(ReturnValue.UPDATED_NEW)
            .build();

        // Perform the update operation
        try {
            return dynamoDbClient.updateItem(request);
        } catch (DynamoDbException e) {
            throw DynamoDbException.builder()
                .message("Failed to update item with all operation types: " +
e.getMessage())
                .cause(e)
                .build();
```

```
        }
    }

    /**
     * Gets the current state of an item.
     *
     * <p>Helper method to retrieve the current state of an item.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to get
     * @return The item or null if not found
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static Map<String, AttributeValue> getItem(
        DynamoDbClient dynamoDbClient, String tableName, Map<String, AttributeValue>
key) {

        // Define the get parameters
        GetItemRequest request =
            GetItemRequest.builder().tableName(tableName).key(key).build();

        // Perform the get operation
        try {
            GetItemResponse response = dynamoDbClient.getItem(request);

            // Return the item if it exists, otherwise null
            return response.item();
        } catch (DynamoDbException e) {
            throw DynamoDbException.builder()
                .message("Failed to get item: " + e.getMessage())
                .cause(e)
                .build();
        }
    }
```

Esempio di utilizzo dell'ordine di aggiornamento delle espressioni con AWS SDK for Java 2.x.

```
    public static void exampleUsage(DynamoDbClient dynamoDbClient, String tableName)
{
        // Example key
        Map<String, AttributeValue> key = new HashMap<>();
```

```
        key.put("ProductId", AttributeValue.builder().s("P12345").build());

        System.out.println("Demonstrating update expression order in DynamoDB");

        try {
            // Example 1: Demonstrate update order effects
            System.out.println("\nExample 1: Demonstrating update order effects");
            Map<String, Object> orderResults =
demonstrateUpdateOrder(dynamoDbClient, tableName, key);

            if ((boolean) orderResults.get("success")) {
                System.out.println("SET first, then ADD:");
                System.out.println("  " + orderResults.get("setFirstResponse"));

                System.out.println("ADD first, then SET:");
                System.out.println("  " + orderResults.get("addFirstResponse"));

                System.out.println("SET with multiple attributes:");
                System.out.println("  " + orderResults.get("multiSetResponse"));

                System.out.println("SET with self-reference:");
                System.out.println("  " +
orderResults.get("selfReferenceResponse"));
            } else {
                System.out.println("Error: " + orderResults.get("error"));
            }

            // Example 2: Update with SET first, then REMOVE
            System.out.println("\nExample 2: Update with SET first, then REMOVE");
            UpdateItemResponse setFirstResponse = updateWithSetFirst(
                dynamoDbClient,
                tableName,
                key,
                "Status",
                AttributeValue.builder().s("Active").build(),
                "OldStatus");

            System.out.println("Updated attributes: " +
setFirstResponse.attributes());

            // Example 3: Update with REMOVE first, then SET
            System.out.println("\nExample 3: Update with REMOVE first, then SET");
            UpdateItemResponse removeFirstResponse = updateWithRemoveFirst(
                dynamoDbClient,
```

```
                tableName,
                key,
                "Status",
                AttributeValue.builder().s("Inactive").build(),
                "OldStatus");

            System.out.println("Updated attributes: " +
removeFirstResponse.attributes());

            // Example 4: Update with all operation types
            System.out.println("\nExample 4: Update with all operation types");
            UpdateItemResponse allOpsResponse =
updateWithAllOperationTypes(dynamoDbClient, tableName, key);

            System.out.println("Updated attributes: " +
allOpsResponse.attributes());

            // Example 5: Get the current state of the item
            System.out.println("\nExample 5: Current state of the item");
            Map<String, AttributeValue> item = getItem(dynamoDbClient, tableName,
key);

            if (item != null) {
                System.out.println("Item: " + item);
            } else {
                System.out.println("Item not found");
            }

            // Explain update expression order
            System.out.println("\nKey points about update expression order in
DynamoDB:");
            System.out.println("1. Update expressions are processed in this order:
SET, REMOVE, ADD, DELETE");
            System.out.println("2. Within each clause, operations are processed from
left to right");
            System.out.println("3. SET operations use the item state before any
updates in the expression");
            System.out.println("4. When an attribute is referenced multiple times,
the first operation wins");
            System.out.println("5. To reference a new value, split the update into
multiple operations");
            System.out.println("6. The order of clauses in the expression doesn't
change the evaluation order");
```

```
            System.out.println("7. For complex updates, consider using multiple
 separate update operations");

        } catch (DynamoDbException e) {
            System.err.println("Error: " + e.getMessage());
            e.printStackTrace();
        }
    }
```

- Per i dettagli sull'API, vedere UpdateItemin AWS SDK for Java 2.x API Reference.

Aggiorna l'impostazione del throughput caldo di una tabella

Il seguente esempio di codice mostra come aggiornare l'impostazione del throughput caldo di una tabella.

SDK per Java 2.x

Aggiorna l'impostazione del throughput caldo su una tabella DynamoDB esistente utilizzando. AWS SDK for Java 2.x

```
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.GlobalSecondaryIndexUpdate;
import
 software.amazon.awssdk.services.dynamodb.model.UpdateGlobalSecondaryIndexAction;
import software.amazon.awssdk.services.dynamodb.model.UpdateTableRequest;
import software.amazon.awssdk.services.dynamodb.model.WarmThroughput;

    public static WarmThroughput buildWarmThroughput(final Long readUnitsPerSecond,
 final Long writeUnitsPerSecond) {
        return WarmThroughput.builder()
            .readUnitsPerSecond(readUnitsPerSecond)
            .writeUnitsPerSecond(writeUnitsPerSecond)
            .build();
    }

    /**
     * Updates a DynamoDB table with warm throughput settings for both the table and
 a global secondary index.
     *
     * @param ddb The DynamoDB client
```

```
    * @param tableName The name of the table to update
    * @param tableReadUnitsPerSecond Read units per second for the table
    * @param tableWriteUnitsPerSecond Write units per second for the table
    * @param globalSecondaryIndexName The name of the global secondary index to
update
    * @param globalSecondaryIndexReadUnitsPerSecond Read units per second for the
GSI
    * @param globalSecondaryIndexWriteUnitsPerSecond Write units per second for the
GSI
    */
   public static void updateDynamoDBTable(
       final DynamoDbClient ddb,
       final String tableName,
       final Long tableReadUnitsPerSecond,
       final Long tableWriteUnitsPerSecond,
       final String globalSecondaryIndexName,
       final Long globalSecondaryIndexReadUnitsPerSecond,
       final Long globalSecondaryIndexWriteUnitsPerSecond) {

       final WarmThroughput tableWarmThroughput =
           buildWarmThroughput(tableReadUnitsPerSecond, tableWriteUnitsPerSecond);
       final WarmThroughput gsiWarmThroughput =
           buildWarmThroughput(globalSecondaryIndexReadUnitsPerSecond,
globalSecondaryIndexWriteUnitsPerSecond);

       final GlobalSecondaryIndexUpdate globalSecondaryIndexUpdate =
GlobalSecondaryIndexUpdate.builder()
           .update(UpdateGlobalSecondaryIndexAction.builder()
               .indexName(globalSecondaryIndexName)
               .warmThroughput(gsiWarmThroughput)
               .build())
           .build();

       final UpdateTableRequest request = UpdateTableRequest.builder()
           .tableName(tableName)
           .globalSecondaryIndexUpdates(globalSecondaryIndexUpdate)
           .warmThroughput(tableWarmThroughput)
           .build();

       try {
           ddb.updateTable(request);
       } catch (DynamoDbException e) {
           System.err.println(e.getMessage());
           throw e;
```

```
        }

        System.out.println(SUCCESS_MESSAGE);
    }
```

- Per i dettagli sull'API, consulta la sezione API UpdateTableReference 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 java.util.HashMap;
import java.util.Map;
import java.util.Optional;

    public UpdateItemResponse updateItemWithTTL(
        final String tableName, final String primaryKeyValue, final String
 sortKeyValue) {
        // 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 + (DAYS_TO_EXPIRE * SECONDS_PER_DAY);

        // Create the key map for the item to update
        final Map<String, AttributeValue> keyMap = new HashMap<>();
        keyMap.put(PRIMARY_KEY_ATTR,
 AttributeValue.builder().s(primaryKeyValue).build());
        keyMap.put(SORT_KEY_ATTR, AttributeValue.builder().s(sortKeyValue).build());
```

```
        // Create the expression attribute values
        final Map<String, AttributeValue> expressionAttributeValues = new
HashMap<>();
        expressionAttributeValues.put(
            ":c", AttributeValue.builder().n(String.valueOf(currentTime)).build());
        expressionAttributeValues.put(
            ":e", AttributeValue.builder().n(String.valueOf(expireDate)).build());

        final UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(keyMap)
            .updateExpression(UPDATE_EXPRESSION)
            .expressionAttributeValues(expressionAttributeValues)
            .build();

        try {
            final UpdateItemResponse response = dynamoDbClient.updateItem(request);
            System.out.println(String.format(SUCCESS_MESSAGE, tableName));
            return response;
        } catch (ResourceNotFoundException e) {
            System.err.format(TABLE_NOT_FOUND_ERROR, tableName);
            throw e;
        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            throw e;
        }
    }
```

- Per i dettagli sull'API, consulta la sezione API UpdateItemReference AWS SDK for Java 2.x .

Utilizzo di un'API Gateway per richiamare una funzione Lambda

Il seguente esempio di codice mostra come creare una AWS Lambda funzione richiamata da Amazon API Gateway.

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

Il seguente esempio di codice mostra come creare una macchina a AWS Step Functions stati che richiama AWS Lambda funzioni in sequenza.

SDK per Java 2.x

Mostra come creare un flusso di lavoro AWS senza server 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

Usa le controoperazioni atomiche

Il seguente esempio di codice mostra come utilizzare le operazioni del contatore atomico in DynamoDB.

- Incrementa i contatori atomicamente utilizzando le operazioni ADD e SET.
- Incrementa in modo sicuro i contatori che potrebbero non esistere.
- Implementa un blocco ottimistico per le operazioni di sportello.

SDK per Java 2.x

Dimostra le controoperazioni atomiche utilizzando. AWS SDK for Java 2.x

```
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.GetItemRequest;
import software.amazon.awssdk.services.dynamodb.model.GetItemResponse;
import software.amazon.awssdk.services.dynamodb.model.ReturnValue;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemRequest;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemResponse;

import java.util.HashMap;
import java.util.Map;

    /**
     * Increments a counter using the ADD operation.
     *
     * <p>This method demonstrates how to use the ADD operation to atomically
     * increment a counter attribute.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param counterName The name of the counter attribute
     * @param incrementValue The value to increment by
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static UpdateItemResponse incrementCounterWithAdd(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        String counterName,
        int incrementValue) {

        // Define the update parameters
```

```
        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .updateExpression("ADD #counterName :increment")
            .expressionAttributeNames(Map.of("#counterName", counterName))
            .expressionAttributeValues(Map.of(
                ":increment",
                AttributeValue.builder().n(String.valueOf(incrementValue)).build()))
            .returnValues(ReturnValue.UPDATED_NEW)
            .build();

        // Perform the update operation
        return dynamoDbClient.updateItem(request);
    }

    /**
     * Increments a counter using the SET operation.
     *
     * <p>This method demonstrates how to use the SET operation with an expression
     * to increment a counter attribute.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param counterName The name of the counter attribute
     * @param incrementValue The value to increment by
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static UpdateItemResponse incrementCounterWithSet(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        String counterName,
        int incrementValue) {

        // Define the update parameters
        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .updateExpression("SET #counterName = #counterName + :increment")
            .expressionAttributeNames(Map.of("#counterName", counterName))
            .expressionAttributeValues(Map.of(
                ":increment",
```

```
                AttributeValue.builder().n(String.valueOf(incrementValue)).build()))
            .returnValues(ReturnValue.UPDATED_NEW)
            .build();

        // Perform the update operation
        return dynamoDbClient.updateItem(request);
    }

    /**
     * Increments a counter safely, handling the case where the counter doesn't
exist yet.
     *
     * <p>This method demonstrates how to use if_not_exists to safely increment a
counter
     * that may not exist yet.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param counterName The name of the counter attribute
     * @param incrementValue The value to increment by
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static UpdateItemResponse incrementCounterSafely(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        String counterName,
        int incrementValue) {

        // Define the update parameters
        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .updateExpression("SET #counterName = if_not_exists(#counterName, :zero)
+ :increment")
            .expressionAttributeNames(Map.of("#counterName", counterName))
            .expressionAttributeValues(Map.of(
                ":increment",

AttributeValue.builder().n(String.valueOf(incrementValue)).build(),
                ":zero", AttributeValue.builder().n("0").build()))
            .returnValues(ReturnValue.UPDATED_NEW)
```

```
            .build();

        // Perform the update operation
        return dynamoDbClient.updateItem(request);
    }

    /**
     * Decrements a counter safely, ensuring it doesn't go below zero.
     *
     * <p>This method demonstrates how to use a condition expression to safely
     * decrement a counter without going below zero.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param counterName The name of the counter attribute
     * @param decrementValue The value to decrement by
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation or if the
counter would go below zero
     */
    public static UpdateItemResponse decrementCounterSafely(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        String counterName,
        int decrementValue) {

        // Define the update parameters
        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .updateExpression("SET #counterName = #counterName - :decrement")
            .conditionExpression("#counterName >= :decrement")
            .expressionAttributeNames(Map.of("#counterName", counterName))
            .expressionAttributeValues(Map.of(
                ":decrement",
                AttributeValue.builder().n(String.valueOf(decrementValue)).build()))
            .returnValues(ReturnValue.UPDATED_NEW)
            .build();

        // Perform the update operation
        return dynamoDbClient.updateItem(request);
    }
```

```
   /**
    * Compares the ADD and SET approaches for incrementing counters.
    *
    * <p>This method demonstrates the differences between using ADD and SET
    * for incrementing counters in DynamoDB.
    *
    * @param dynamoDbClient The DynamoDB client
    * @param tableName The name of the DynamoDB table
    * @param key The key of the item to update
    * @return Map containing the comparison results
    */
   public static Map<String, Object> compareAddVsSet(
       DynamoDbClient dynamoDbClient, String tableName, Map<String, AttributeValue>
key) {

       Map<String, Object> results = new HashMap<>();

       try {
           // Reset counters to ensure a fair comparison
           UpdateItemRequest resetRequest = UpdateItemRequest.builder()
               .tableName(tableName)
               .key(key)
               .updateExpression("SET AddCounter = :zero, SetCounter = :zero")
               .expressionAttributeValues(
                   Map.of(":zero", AttributeValue.builder().n("0").build()))
               .build();

           dynamoDbClient.updateItem(resetRequest);

           // Increment with ADD
           long addStartTime = System.nanoTime();
           UpdateItemResponse addResponse = incrementCounterWithAdd(dynamoDbClient,
tableName, key, "AddCounter", 1);
           long addEndTime = System.nanoTime();
           long addDuration = addEndTime - addStartTime;

           // Increment with SET
           long setStartTime = System.nanoTime();
           UpdateItemResponse setResponse = incrementCounterWithSet(dynamoDbClient,
tableName, key, "SetCounter", 1);
           long setEndTime = System.nanoTime();
           long setDuration = setEndTime - setStartTime;
```

```
            // Record results
            results.put("addResponse", addResponse);
            results.put("setResponse", setResponse);
            results.put("addDuration", addDuration);
            results.put("setDuration", setDuration);
            results.put("success", true);

        } catch (DynamoDbException e) {
            results.put("success", false);
            results.put("error", e.getMessage());
        }

        return results;
    }

    /**
     * Gets the current value of a counter attribute.
     *
     * <p>Helper method to retrieve the current value of a counter attribute.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to get
     * @param counterName The name of the counter attribute
     * @return The counter value or null if not found
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static Integer getCounterValue(
        DynamoDbClient dynamoDbClient, String tableName, Map<String, AttributeValue>
key, String counterName) {

        // Define the get parameters
        GetItemRequest request = GetItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .projectionExpression(counterName)
            .build();

        // Perform the get operation
        GetItemResponse response = dynamoDbClient.getItem(request);

        // Return the counter value if it exists, otherwise null
        if (response.item() != null && response.item().containsKey(counterName)) {
            return Integer.parseInt(response.item().get(counterName).n());
```

```
        }

        return null;
    }
```

Esempio di utilizzo delle operazioni di conteggio atomico con. AWS SDK for Java 2.x

```
    public static void exampleUsage(DynamoDbClient dynamoDbClient, String tableName)
{
        // Example key
        Map<String, AttributeValue> key = new HashMap<>();
        key.put("ProductId", AttributeValue.builder().s("P12345").build());

        System.out.println("Demonstrating atomic counter operations in DynamoDB");

        try {
            // Example 1: Increment a counter using ADD
            System.out.println("\nExample 1: Incrementing a counter using ADD");
            UpdateItemResponse addResponse = incrementCounterWithAdd(dynamoDbClient,
tableName, key, "ViewCount", 1);

            System.out.println("Updated counter: " + addResponse.attributes());

            // Example 2: Increment a counter using SET
            System.out.println("\nExample 2: Incrementing a counter using SET");
            UpdateItemResponse setResponse = incrementCounterWithSet(dynamoDbClient,
tableName, key, "LikeCount", 1);

            System.out.println("Updated counter: " + setResponse.attributes());

            // Example 3: Increment a counter safely
            System.out.println("\nExample 3: Incrementing a counter safely");
            UpdateItemResponse safeResponse = incrementCounterSafely(dynamoDbClient,
tableName, key, "ShareCount", 1);

            System.out.println("Updated counter: " + safeResponse.attributes());

            // Example 4: Decrement a counter safely
            System.out.println("\nExample 4: Decrementing a counter safely");
            try {
                UpdateItemResponse decrementResponse =
```

```
                    decrementCounterSafely(dynamoDbClient, tableName, key,
"InventoryCount", 1);

                System.out.println("Updated counter: " +
decrementResponse.attributes());
            } catch (DynamoDbException e) {
                if (e.getMessage().contains("ConditionalCheckFailed")) {
                    System.out.println("Cannot decrement counter below zero");
                } else {
                    throw e;
                }
            }

            // Example 5: Compare ADD vs SET
            System.out.println("\nExample 5: Comparing ADD vs SET");
            Map<String, Object> comparison = compareAddVsSet(dynamoDbClient,
tableName, key);

            if ((boolean) comparison.get("success")) {
                System.out.println("ADD duration: " + comparison.get("addDuration")
+ " ns");
                System.out.println("SET duration: " + comparison.get("setDuration")
+ " ns");
                System.out.println("ADD response: " +
comparison.get("addResponse"));
                System.out.println("SET response: " +
comparison.get("setResponse"));
            } else {
                System.out.println("Comparison failed: " + comparison.get("error"));
            }

            // Explain atomic counter operations
            System.out.println("\nKey points about DynamoDB atomic counter
operations:");
            System.out.println("1. Both ADD and SET can be used for atomic
counters");
            System.out.println("2. ADD is more concise for simple increments");
            System.out.println("3. SET with an expression is more flexible for
complex operations");
            System.out.println("4. Use if_not_exists to handle the case where the
counter doesn't exist yet");
            System.out.println("5. Use condition expressions to prevent counters
from going below zero");
```

```
            System.out.println("6. Atomic operations are guaranteed to be isolated
from other writes");
            System.out.println("7. ADD can only be used with number and set data
types");

        } catch (DynamoDbException e) {
            System.err.println("Error: " + e.getMessage());
            e.printStackTrace();
        }
    }
```

- Per i dettagli sull'API, vedere UpdateItemin AWS SDK for Java 2.x API Reference.

Usa operazioni condizionali

Il seguente esempio di codice mostra come utilizzare le operazioni condizionali in DynamoDB.

- Implementa scritture condizionali per evitare la sovrascrittura dei dati.
- Utilizza le espressioni condizionali per applicare le regole aziendali.
- Gestisci con garbo gli errori dei controlli condizionali.

SDK per Java 2.x

Dimostra le operazioni condizionali utilizzando. AWS SDK for Java 2.x

```
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import
 software.amazon.awssdk.services.dynamodb.model.ConditionalCheckFailedException;
import software.amazon.awssdk.services.dynamodb.model.DeleteItemRequest;
import software.amazon.awssdk.services.dynamodb.model.DeleteItemResponse;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.GetItemRequest;
import software.amazon.awssdk.services.dynamodb.model.GetItemResponse;
import software.amazon.awssdk.services.dynamodb.model.ReturnValue;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemRequest;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemResponse;

import java.util.HashMap;
import java.util.Map;
```

```
   /**
    * Performs a conditional update on an item.
    *
    * <p>This method demonstrates how to use a condition expression to update an
item
    * only if a specific condition is met.
    *
    * @param dynamoDbClient The DynamoDB client
    * @param tableName The name of the DynamoDB table
    * @param key The key of the item to update
    * @param conditionAttribute The attribute to check in the condition
    * @param conditionValue The value to compare against
    * @param updateAttribute The attribute to update
    * @param updateValue The new value to set
    * @return Map containing the operation result and status
    */
   public static Map<String, Object> conditionalUpdate(
       DynamoDbClient dynamoDbClient,
       String tableName,
       Map<String, AttributeValue> key,
       String conditionAttribute,
       AttributeValue conditionValue,
       String updateAttribute,
       AttributeValue updateValue) {

       Map<String, Object> result = new HashMap<>();

       try {
           // Define the update parameters
           UpdateItemRequest request = UpdateItemRequest.builder()
               .tableName(tableName)
               .key(key)
               .updateExpression("SET #updateAttr = :updateVal")
               .conditionExpression("#condAttr = :condVal")
               .expressionAttributeNames(Map.of(
                   "#condAttr", conditionAttribute,
                   "#updateAttr", updateAttribute))
               .expressionAttributeValues(Map.of(
                   ":condVal", conditionValue,
                   ":updateVal", updateValue))
               .returnValues(ReturnValue.UPDATED_NEW)
               .build();
```

```
            // Perform the update operation
            UpdateItemResponse response = dynamoDbClient.updateItem(request);

            // Record success result
            result.put("success", true);
            result.put("message", "Condition was met and update was performed");
            result.put("attributes", response.attributes());

        } catch (ConditionalCheckFailedException e) {
            // Record failure due to condition not being met
            result.put("success", false);
            result.put("message", "Condition was not met, update was not
performed");
            result.put("error", "ConditionalCheckFailedException");

        } catch (DynamoDbException e) {
            // Record failure due to other errors
            result.put("success", false);
            result.put("message", "Error occurred: " + e.getMessage());
            result.put("error", e.getClass().getSimpleName());
        }

        return result;
    }

    /**
     * Performs a conditional delete on an item.
     *
     * <p>This method demonstrates how to use a condition expression to delete an
item
     * only if a specific condition is met.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to delete
     * @param conditionAttribute The attribute to check in the condition
     * @param conditionValue The value to compare against
     * @return Map containing the operation result and status
     */
    public static Map<String, Object> conditionalDelete(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        String conditionAttribute,
```

```
        AttributeValue conditionValue) {

        Map<String, Object> result = new HashMap<>();

        try {
            // Define the delete parameters
            DeleteItemRequest request = DeleteItemRequest.builder()
                .tableName(tableName)
                .key(key)
                .conditionExpression("#condAttr = :condVal")
                .expressionAttributeNames(Map.of("#condAttr", conditionAttribute))
                .expressionAttributeValues(Map.of(":condVal", conditionValue))
                .returnValues(ReturnValue.ALL_OLD)
                .build();

            // Perform the delete operation
            DeleteItemResponse response = dynamoDbClient.deleteItem(request);

            // Record success result
            result.put("success", true);
            result.put("message", "Condition was met and delete was performed");
            result.put("attributes", response.attributes());

        } catch (ConditionalCheckFailedException e) {
            // Record failure due to condition not being met
            result.put("success", false);
            result.put("message", "Condition was not met, delete was not
performed");
            result.put("error", "ConditionalCheckFailedException");

        } catch (DynamoDbException e) {
            // Record failure due to other errors
            result.put("success", false);
            result.put("message", "Error occurred: " + e.getMessage());
            result.put("error", e.getClass().getSimpleName());
        }

        return result;
    }

    /**
     * Demonstrates optimistic locking using a version attribute.
     *
     * <p>This method shows how to implement optimistic locking by using a version
```

```
     * attribute that is incremented with each update.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param versionAttribute The name of the version attribute
     * @return Map containing the operation result
     */
    public static Map<String, Object> optimisticLockingExample(
        DynamoDbClient dynamoDbClient, String tableName, Map<String, AttributeValue>
 key, String versionAttribute) {

        Map<String, Object> result = new HashMap<>();

        try {
            // Get the current version of the item
            GetItemRequest getRequest = GetItemRequest.builder()
                .tableName(tableName)
                .key(key)
                .projectionExpression(versionAttribute)
                .build();

            GetItemResponse getResponse = dynamoDbClient.getItem(getRequest);

            // Check if the item exists
            if (getResponse.item() == null || !
getResponse.item().containsKey(versionAttribute)) {
                // Item doesn't exist or doesn't have a version attribute
                // Initialize with version 1
                UpdateItemRequest initRequest = UpdateItemRequest.builder()
                    .tableName(tableName)
                    .key(key)
                    .updateExpression("SET #verAttr = :newVer, #dataAttr = :data")
                    .expressionAttributeNames(Map.of("#verAttr", versionAttribute,
 "#dataAttr", "Data"))
                    .expressionAttributeValues(Map.of(
                        ":newVer", AttributeValue.builder().n("1").build(),
                        ":data", AttributeValue.builder().s("Initial
 data").build()))
                    .returnValues(ReturnValue.UPDATED_NEW)
                    .build();

                UpdateItemResponse initResponse =
 dynamoDbClient.updateItem(initRequest);
```

```
                result.put("operation", "initialize");
                result.put("success", true);
                result.put("attributes", initResponse.attributes());

                return result;
            }

            // Get the current version number
            int currentVersion =
                Integer.parseInt(getResponse.item().get(versionAttribute).n());
            int newVersion = currentVersion + 1;

            // Update the item with a condition on the version
            UpdateItemRequest updateRequest = UpdateItemRequest.builder()
                .tableName(tableName)
                .key(key)
                .updateExpression("SET #verAttr = :newVer, #dataAttr = :newData")
                .conditionExpression("#verAttr = :curVer")
                .expressionAttributeNames(Map.of("#verAttr", versionAttribute,
"#dataAttr", "Data"))
                .expressionAttributeValues(Map.of(
                    ":curVer",
                        AttributeValue.builder()
                            .n(String.valueOf(currentVersion))
                            .build(),
                    ":newVer",

AttributeValue.builder().n(String.valueOf(newVersion)).build(),
                    ":newData",
                        AttributeValue.builder()
                            .s("Updated data at version " + newVersion)
                            .build()))
                .returnValues(ReturnValue.UPDATED_NEW)
                .build();

            UpdateItemResponse updateResponse =
dynamoDbClient.updateItem(updateRequest);

            // Record success result
            result.put("operation", "update");
            result.put("success", true);
            result.put("oldVersion", currentVersion);
            result.put("newVersion", newVersion);
```

```
            result.put("attributes", updateResponse.attributes());

        } catch (ConditionalCheckFailedException e) {
            // Record failure due to version mismatch
            result.put("operation", "update");
            result.put("success", false);
            result.put("message", "Version mismatch, another process may have
updated the item");
            result.put("error", "ConditionalCheckFailedException");

        } catch (DynamoDbException e) {
            // Record failure due to other errors
            result.put("operation", "update");
            result.put("success", false);
            result.put("message", "Error occurred: " + e.getMessage());
            result.put("error", e.getClass().getSimpleName());
        }

        return result;
    }

    /**
     * Performs a conditional update with multiple conditions.
     *
     * <p>This method demonstrates how to use multiple conditions in a condition
expression.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param conditions Map of attribute names to values for conditions
     * @param updateAttribute The attribute to update
     * @param updateValue The new value to set
     * @return Map containing the operation result and status
     */
    public static Map<String, Object> conditionalUpdateWithMultipleConditions(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        Map<String, AttributeValue> conditions,
        String updateAttribute,
        AttributeValue updateValue) {

        Map<String, Object> result = new HashMap<>();
```

```
        try {
            // Build the condition expression and attribute names/values
            StringBuilder conditionExpression = new StringBuilder();
            Map<String, String> expressionAttributeNames = new HashMap<>();
            Map<String, AttributeValue> expressionAttributeValues = new HashMap<>();

            // Add update attribute
            expressionAttributeNames.put("#updateAttr", updateAttribute);
            expressionAttributeValues.put(":updateVal", updateValue);

            // Add conditions
            int i = 0;
            for (Map.Entry<String, AttributeValue> condition :
conditions.entrySet()) {
                String attrName = condition.getKey();
                AttributeValue attrValue = condition.getValue();

                String nameKey = "#cond" + i;
                String valueKey = ":val" + i;

                expressionAttributeNames.put(nameKey, attrName);
                expressionAttributeValues.put(valueKey, attrValue);

                // Add AND between conditions (except for the first one)
                if (i > 0) {
                    conditionExpression.append(" AND ");
                }

                conditionExpression.append(nameKey).append(" = ").append(valueKey);
                i++;
            }

            // Define the update parameters
            UpdateItemRequest request = UpdateItemRequest.builder()
                .tableName(tableName)
                .key(key)
                .updateExpression("SET #updateAttr = :updateVal")
                .conditionExpression(conditionExpression.toString())
                .expressionAttributeNames(expressionAttributeNames)
                .expressionAttributeValues(expressionAttributeValues)
                .returnValues(ReturnValue.UPDATED_NEW)
                .build();
```

```
            // Perform the update operation
            UpdateItemResponse response = dynamoDbClient.updateItem(request);

            // Record success result
            result.put("success", true);
            result.put("message", "All conditions were met and update was
performed");
            result.put("attributes", response.attributes());

        } catch (ConditionalCheckFailedException e) {
            // Record failure due to condition not being met
            result.put("success", false);
            result.put("message", "One or more conditions were not met, update was
not performed");
            result.put("error", "ConditionalCheckFailedException");

        } catch (DynamoDbException e) {
            // Record failure due to other errors
            result.put("success", false);
            result.put("message", "Error occurred: " + e.getMessage());
            result.put("error", e.getClass().getSimpleName());
        }

        return result;
    }
```

Esempio di utilizzo di operazioni condizionali con. AWS SDK for Java 2.x

```
    public static void exampleUsage(DynamoDbClient dynamoDbClient, String tableName)
{
        // Example key
        Map<String, AttributeValue> key = new HashMap<>();
        key.put("ProductId", AttributeValue.builder().s("P12345").build());

        System.out.println("Demonstrating conditional operations in DynamoDB");

        try {
            // Example 1: Conditional update
            System.out.println("\nExample 1: Conditional update");
            Map<String, Object> updateResult = conditionalUpdate(
                dynamoDbClient,
                tableName,
```

```
                key,
                "InStock",
                AttributeValue.builder().bool(true).build(),
                "Status",
                AttributeValue.builder().s("Available").build());

            System.out.println("Update result: " + updateResult.get("message"));
            if ((boolean) updateResult.get("success")) {
                System.out.println("Updated attributes: " +
updateResult.get("attributes"));
            }

            // Example 2: Conditional delete
            System.out.println("\nExample 2: Conditional delete");
            Map<String, Object> deleteResult = conditionalDelete(
                dynamoDbClient,
                tableName,
                key,
                "Status",
                AttributeValue.builder().s("Discontinued").build());

            System.out.println("Delete result: " + deleteResult.get("message"));
            if ((boolean) deleteResult.get("success")) {
                System.out.println("Deleted item: " +
deleteResult.get("attributes"));
            }

            // Example 3: Optimistic locking
            System.out.println("\nExample 3: Optimistic locking");
            Map<String, Object> lockingResult =
optimisticLockingExample(dynamoDbClient, tableName, key, "Version");

            System.out.println("Optimistic locking result:");
            System.out.println("  Operation: " + lockingResult.get("operation"));
            System.out.println("  Success: " + lockingResult.get("success"));
            if (lockingResult.get("operation").equals("update") && (boolean)
lockingResult.get("success")) {
                System.out.println("  Old version: " +
lockingResult.get("oldVersion"));
                System.out.println("  New version: " +
lockingResult.get("newVersion"));
            }
            System.out.println("  Attributes: " + lockingResult.get("attributes"));
```

```
            // Example 4: Multiple conditions
            System.out.println("\nExample 4: Multiple conditions");
            Map<String, AttributeValue> conditions = new HashMap<>();
            conditions.put("Price", AttributeValue.builder().n("199.99").build());
            conditions.put("Category",
AttributeValue.builder().s("Electronics").build());

            Map<String, Object> multiConditionResult =
conditionalUpdateWithMultipleConditions(
                dynamoDbClient,
                tableName,
                key,
                conditions,
                "OnSale",
                AttributeValue.builder().bool(true).build());

            System.out.println("Multiple conditions result: " +
multiConditionResult.get("message"));
            if ((boolean) multiConditionResult.get("success")) {
                System.out.println("Updated attributes: " +
multiConditionResult.get("attributes"));
            }

            // Explain conditional operations
            System.out.println("\nKey points about DynamoDB conditional
operations:");
            System.out.println("1. Conditional operations only succeed if the
condition is met");
            System.out.println("2. ConditionalCheckFailedException is thrown when
the condition fails");
            System.out.println("3. No changes are made to the item if the condition
fails");
            System.out.println("4. Conditions can be used with update, delete, and
put operations");
            System.out.println("5. Multiple conditions can be combined with AND and
OR");
            System.out.println("6. Optimistic locking can be implemented using a
version attribute");
            System.out.println(
                "7. Conditional operations consume the same amount of write capacity
whether they succeed or fail");

        } catch (DynamoDbException e) {
            System.err.println("Error: " + 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 .
  - DeleteItem
  - PutItem
  - UpdateItem

Usa i nomi degli attributi di espressione

Il seguente esempio di codice mostra come utilizzare i nomi degli attributi di espressione in DynamoDB.

- Lavora con parole riservate nelle espressioni DynamoDB.
- Usa i segnaposto per i nomi degli attributi di espressione.
- Gestisci i caratteri speciali nei nomi degli attributi.

SDK per Java 2.x

Dimostra i nomi degli attributi di espressione usando AWS SDK for Java 2.x.

```
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.ScanRequest;
import software.amazon.awssdk.services.dynamodb.model.ScanResponse;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemRequest;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemResponse;

import java.util.Arrays;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

    /**
```

```
     * Updates an attribute that is a reserved word in DynamoDB.
     *
     * <p>This method demonstrates how to use expression attribute names to update
     * attributes that are reserved words in DynamoDB.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param reservedWordAttribute The reserved word attribute to update
     * @param value The value to set
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static UpdateItemResponse updateReservedWordAttribute(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        String reservedWordAttribute,
        AttributeValue value) {

        // Define the update parameters using expression attribute names
        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .updateExpression("SET #attr = :value")
            .expressionAttributeNames(Map.of("#attr", reservedWordAttribute))
            .expressionAttributeValues(Map.of(":value", value))
            .returnValues("UPDATED_NEW")
            .build();

        // Perform the update operation
        return dynamoDbClient.updateItem(request);
    }

    /**
     * Updates an attribute that contains special characters.
     *
     * <p>This method demonstrates how to use expression attribute names to update
     * attributes that contain special characters.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param specialCharAttribute The attribute with special characters to update
```

```
     * @param value The value to set
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static UpdateItemResponse updateSpecialCharacterAttribute(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        String specialCharAttribute,
        AttributeValue value) {

        // Define the update parameters using expression attribute names
        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .updateExpression("SET #attr = :value")
            .expressionAttributeNames(Map.of("#attr", specialCharAttribute))
            .expressionAttributeValues(Map.of(":value", value))
            .returnValues("UPDATED_NEW")
            .build();

        // Perform the update operation
        return dynamoDbClient.updateItem(request);
    }

    /**
     * Queries items using an attribute that is a reserved word.
     *
     * <p>This method demonstrates how to use expression attribute names in a query
     * when the attribute is a reserved word.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param partitionKeyName The name of the partition key attribute
     * @param partitionKeyValue The value of the partition key
     * @param reservedWordAttribute The reserved word attribute to filter on
     * @param value The value to compare against
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static QueryResponse queryWithReservedWordAttribute(
        DynamoDbClient dynamoDbClient,
        String tableName,
        String partitionKeyName,
```

```
        AttributeValue partitionKeyValue,
        String reservedWordAttribute,
        AttributeValue value) {

        // Define the query parameters using expression attribute names
        Map<String, String> expressionAttributeNames = new HashMap<>();
        expressionAttributeNames.put("#pkName", partitionKeyName);
        expressionAttributeNames.put("#attr", reservedWordAttribute);

        Map<String, AttributeValue> expressionAttributeValues = new HashMap<>();
        expressionAttributeValues.put(":pkValue", partitionKeyValue);
        expressionAttributeValues.put(":value", value);

        QueryRequest request = QueryRequest.builder()
            .tableName(tableName)
            .keyConditionExpression("#pkName = :pkValue")
            .filterExpression("#attr = :value")
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .build();

        // Perform the query operation
        return dynamoDbClient.query(request);
    }

    /**
     * Updates a nested attribute with a path that contains reserved words.
     *
     * <p>This method demonstrates how to use expression attribute names to update
     * nested attributes where the path contains reserved words.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param attributePath The path to the nested attribute as an array
     * @param value The value to set
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static UpdateItemResponse updateNestedReservedWordAttribute(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        List<String> attributePath,
```

```
        AttributeValue value) {

        // Create expression attribute names for each part of the path
        Map<String, String> expressionAttributeNames = new HashMap<>();
        for (int i = 0; i < attributePath.size(); i++) {
            expressionAttributeNames.put("#attr" + i, attributePath.get(i));
        }

        // Build the attribute path using the expression attribute names
        StringBuilder attributePathExpression = new StringBuilder();
        for (int i = 0; i < attributePath.size(); i++) {
            if (i > 0) {
                attributePathExpression.append(".");
            }
            attributePathExpression.append("#attr").append(i);
        }

        // Define the update parameters
        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .updateExpression("SET " + attributePathExpression.toString() + "
= :value")
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(Map.of(":value", value))
            .returnValues("UPDATED_NEW")
            .build();

        // Perform the update operation
        return dynamoDbClient.updateItem(request);
    }

    /**
     * Scans a table with multiple attribute name placeholders.
     *
     * <p>This method demonstrates how to use multiple expression attribute names
     * in a complex filter expression.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param filters Object mapping attribute names to filter values
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
```

```
    public static ScanResponse scanWithMultipleAttributeNames(
        DynamoDbClient dynamoDbClient, String tableName, Map<String, AttributeValue>
filters) {

        // Create expression attribute names and values
        Map<String, String> expressionAttributeNames = new HashMap<>();
        Map<String, AttributeValue> expressionAttributeValues = new HashMap<>();
        StringBuilder filterExpression = new StringBuilder();

        // Build the filter expression
        int index = 0;
        for (Map.Entry<String, AttributeValue> entry : filters.entrySet()) {
            String attrName = entry.getKey();
            AttributeValue attrValue = entry.getValue();

            String nameKey = "#attr" + index;
            String valueKey = ":val" + index;

            expressionAttributeNames.put(nameKey, attrName);
            expressionAttributeValues.put(valueKey, attrValue);

            // Add AND between conditions (except for the first one)
            if (index > 0) {
                filterExpression.append(" AND ");
            }

            filterExpression.append(nameKey).append(" = ").append(valueKey);
            index++;
        }

        // Define the scan parameters
        ScanRequest request = ScanRequest.builder()
            .tableName(tableName)
            .filterExpression(filterExpression.toString())
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .build();

        // Perform the scan operation
        return dynamoDbClient.scan(request);
    }
```

Esempio di utilizzo dei nomi degli attributi di espressione con AWS SDK for Java 2.x.

```
   public static void exampleUsage(DynamoDbClient dynamoDbClient, String tableName)
{
       // Example key
       Map<String, AttributeValue> key = new HashMap<>();
       key.put("ProductId", AttributeValue.builder().s("P12345").build());

       System.out.println("Demonstrating expression attribute names in DynamoDB");

       try {
           // Example 1: Update an attribute that is a reserved word
           System.out.println("\nExample 1: Updating an attribute that is a
reserved word");
           UpdateItemResponse response1 = updateReservedWordAttribute(
               dynamoDbClient,
               tableName,
               key,
               "Size", // "SIZE" is a reserved word in DynamoDB
               AttributeValue.builder().s("Large").build());

           System.out.println("Updated attribute: " + response1.attributes());

           // Example 2: Update an attribute with special characters
           System.out.println("\nExample 2: Updating an attribute with special
characters");
           UpdateItemResponse response2 = updateSpecialCharacterAttribute(
               dynamoDbClient,
               tableName,
               key,
               "Product-Type", // Contains a hyphen, which is a special character
               AttributeValue.builder().s("Electronics").build());

           System.out.println("Updated attribute: " + response2.attributes());

           // Example 3: Query with a reserved word attribute
           System.out.println("\nExample 3: Querying with a reserved word
attribute");
           QueryResponse response3 = queryWithReservedWordAttribute(
               dynamoDbClient,
               tableName,
               "Category",
               AttributeValue.builder().s("Electronics").build(),
               "Count", // "COUNT" is a reserved word in DynamoDB
```

```
                AttributeValue.builder().n("10").build());

            System.out.println("Found " + response3.count() + " items");

            // Example 4: Update a nested attribute with reserved words in the path
            System.out.println("\nExample 4: Updating a nested attribute with
reserved words in the path");
            UpdateItemResponse response4 = updateNestedReservedWordAttribute(
                dynamoDbClient,
                tableName,
                key,
                Arrays.asList("Dimensions", "Size", "Height"), // "SIZE" is a
reserved word
                AttributeValue.builder().n("30").build());

            System.out.println("Updated nested attribute: " +
response4.attributes());

            // Example 5: Scan with multiple attribute name placeholders
            System.out.println("\nExample 5: Scanning with multiple attribute name
placeholders");
            Map<String, AttributeValue> filters = new HashMap<>();
            filters.put("Size", AttributeValue.builder().s("Large").build());
            filters.put("Count", AttributeValue.builder().n("10").build());
            filters.put(
                "Product-Type", AttributeValue.builder().s("Electronics").build());

            ScanResponse response5 = scanWithMultipleAttributeNames(dynamoDbClient,
tableName, filters);

            System.out.println("Found " + response5.count() + " items");

            // Show some common reserved words
            System.out.println("\nSome common DynamoDB reserved words:");
            List<String> commonReservedWords = getDynamoDBReservedWords();
            System.out.println(String.join(", ", commonReservedWords));

            // Explain expression attribute names
            System.out.println("\nKey points about expression attribute names:");
            System.out.println("1. Use expression attribute names (#name) for
reserved words");
            System.out.println("2. Use expression attribute names for attributes
with special characters");
            System.out.println(
```

```
                "3. Special characters include: spaces, hyphens, dots, and other
non-alphanumeric characters");
            System.out.println("4. Expression attribute names are required for
nested attributes with reserved words");
            System.out.println("5. You can use multiple expression attribute names
in a single expression");
            System.out.println("6. Expression attribute names are case-sensitive");
            System.out.println("7. Expression attribute names are only used in
expressions, not in the actual data");

        } catch (DynamoDbException e) {
            System.err.println("Error: " + 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 .
  - Query
  - UpdateItem

Utilizzo degli eventi pianificati per richiamare una funzione Lambda

Il seguente esempio di codice mostra come creare una AWS Lambda funzione richiamata da un evento EventBridge pianificato di Amazon.

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

- CloudWatch Registri
- DynamoDB
- EventBridge
- Lambda
- Amazon SNS

Esempi serverless

Richiamare una funzione Lambda da un trigger DynamoDB

Il seguente esempio di codice mostra come implementare una funzione Lambda che riceve un evento attivato dalla ricezione di record da un flusso DynamoDB. La funzione recupera il payload DocumentDB 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 DynamoDB con Lambda tramite Java.

```
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.DynamodbEvent.DynamodbStreamRecord;
import com.google.gson.Gson;
import com.google.gson.GsonBuilder;

public class example implements RequestHandler<DynamodbEvent, Void> {

    private static final Gson GSON = new GsonBuilder().setPrettyPrinting().create();

    @Override
    public Void handleRequest(DynamodbEvent event, Context context) {
```

```
        System.out.println(GSON.toJson(event));
        event.getRecords().forEach(this::logDynamoDBRecord);
        return null;
    }

    private void logDynamoDBRecord(DynamodbStreamRecord record) {
        System.out.println(record.getEventID());
        System.out.println(record.getEventName());
        System.out.println("DynamoDB Record: " + GSON.toJson(record.getDynamodb()));
    }
}
```

Segnalazione di errori di elementi batch per 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 di errori di elementi batch di DynamoDB 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.DynamodbEvent;
import com.amazonaws.services.lambda.runtime.events.StreamsEventResponse;
import com.amazonaws.services.lambda.runtime.events.models.dynamodb.StreamRecord;

import java.util.ArrayList;
import java.util.List;

public class ProcessDynamodbRecords implements RequestHandler<DynamodbEvent,
 StreamsEventResponse> {
```

```
    @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();
    }
}
```

AWS contributi della comunità

Crea e testa un'applicazione serverless

Il seguente esempio di codice mostra come creare e testare un'applicazione serverless utilizzando API Gateway con Lambda e DynamoDB

SDK per Java 2.x

Mostra come creare e testare un'applicazione serverless composta da un API Gateway con Lambda e DynamoDB utilizzando Java SDK.

Per il codice sorgente completo e le istruzioni su come configurarlo ed eseguirlo, consulta l'esempio completo su. GitHub

Servizi utilizzati in questo esempio

- API Gateway
- DynamoDB
- Lambda

EC2 Esempi di Amazon che utilizzano 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 Amazon EC2.

Le nozioni di base sono esempi di codice che mostrano come eseguire le operazioni essenziali all'interno di un servizio.

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.

Gli scenari sono esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio o combinate con altri Servizi AWS.

Ogni esempio include un collegamento al codice sorgente completo, dove puoi trovare istruzioni su come configurare ed eseguire il codice nel contesto.

Nozioni di base

Ciao Amazon EC2

I seguenti esempi di codice mostrano come iniziare a usare 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.

```
   /**
    * Asynchronously describes the security groups for the specified group ID.
    *
    * @param groupName the name of the security group to describe
    * @return a {@link CompletableFuture} that represents the asynchronous
operation
    *         of describing the security groups. The future will complete with a
    *         {@link DescribeSecurityGroupsResponse} object that contains the
    *         security group information.
    */
   public CompletableFuture<String> describeSecurityGroupArnByNameAsync(String
groupName) {
       DescribeSecurityGroupsRequest request =
DescribeSecurityGroupsRequest.builder()
           .groupNames(groupName)
           .build();

       DescribeSecurityGroupsPublisher paginator =
getAsyncClient().describeSecurityGroupsPaginator(request);
       AtomicReference<String> groupIdRef = new AtomicReference<>();
       return paginator.subscribe(response -> {
           response.securityGroups().stream()
               .filter(securityGroup ->
securityGroup.groupName().equals(groupName))
               .findFirst()
               .ifPresent(securityGroup ->
groupIdRef.set(securityGroup.groupId()));
       }).thenApply(v -> {
           String groupId = groupIdRef.get();
           if (groupId == null) {
               throw new RuntimeException("No security group found with the name: "
+ groupName);
           }
```

```
            return groupId;
        }).exceptionally(ex -> {
            logger.info("Failed to describe security group: " + ex.getMessage());
            throw new RuntimeException("Failed to describe security group", ex);
        });
    }
```

- Per i dettagli sull'API, consulta la DescribeSecurityGroupssezione AWS SDK for Java 2.x API Reference.

Argomenti

- Nozioni di base
- Azioni
- Scenari

Nozioni di base

Informazioni di base

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 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 al prompt dei comandi.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.ec2.model.CreateKeyPairResponse;
import software.amazon.awssdk.services.ec2.model.DeleteKeyPairResponse;
import software.amazon.awssdk.services.ec2.model.DescribeKeyPairsResponse;
import software.amazon.awssdk.services.ec2.model.DisassociateAddressResponse;
import software.amazon.awssdk.services.ec2.model.Ec2Exception;
import software.amazon.awssdk.services.ec2.model.ReleaseAddressResponse;
import software.amazon.awssdk.services.ssm.model.GetParametersByPathResponse;
import software.amazon.awssdk.services.ssm.model.Parameter;

import java.net.InetAddress;
import java.net.UnknownHostException;
import java.util.List;
import java.util.Scanner;
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.CompletionException;

/**
 * Before running this Java (v2) code example, set up your development
 * environment, including 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 additional 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", "-");
    private static final Logger logger = LoggerFactory.getLogger(EC2Scenario.class);
    public static void main(String[] args) throws InterruptedException,
 UnknownHostException {

        logger.info("""
            Usage:
               <keyName> <fileName> <groupName> <groupDesc>

            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
            """);

        Scanner scanner = new Scanner(System.in);
        EC2Actions ec2Actions = new EC2Actions();

        String keyName = "TestKeyPair7" ;
        String fileName = "ec2Key.pem";
        String groupName = "TestSecGroup7" ;
        String groupDesc = "Test Group" ;
        String vpcId = ec2Actions.describeFirstEC2VpcAsync().join().vpcId();
        InetAddress localAddress = InetAddress.getLocalHost();
        String myIpAddress = localAddress.getHostAddress();

        logger.info("""
            Amazon Elastic Compute Cloud (EC2) is a web service that provides
 secure, resizable compute
            capacity in the cloud. It allows developers and organizations to easily
 launch and manage
            virtual server instances, known as EC2 instances, to run their
 applications.
```

```
            EC2 provides a wide range of instance types, each with different
 compute, memory,
            and storage capabilities, to meet the diverse needs of various
 workloads. Developers
            can choose the appropriate instance type based on their application's
 requirements,
            such as high-performance computing, memory-intensive tasks, or GPU-
accelerated workloads.

            The `Ec2AsyncClient` interface in the AWS SDK for Java 2.x provides a
 set of methods to
            programmatically interact with the Amazon EC2 service. This allows
 developers to
            automate the provisioning, management, and monitoring of EC2 instances
 as part of their
            application deployment pipelines. With EC2, teams can focus on building
 and deploying
            their applications without having to worry about the underlying
 infrastructure
            required to host and manage physical servers.

            This scenario walks you through how to perform key operations for this
 service.
            Let's get started...
            """);

        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("1. Create an RSA key pair and save the private key material as
 a .pem file.");
        logger.info("""
            An RSA key pair for Amazon EC2 is a security mechanism used to
 authenticate and secure
            access to your EC2 instances. It consists of a public key and a private
 key,
            which are generated as a pair.
            """);
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<CreateKeyPairResponse> future =
 ec2Actions.createKeyPairAsync(keyName, fileName);
            CreateKeyPairResponse response = future.join();
```

```
            logger.info("Key Pair successfully created. Key Fingerprint: " +
response.keyFingerprint());

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof Ec2Exception ec2Ex) {
                if (ec2Ex.getMessage().contains("already exists")) {
                    // Key pair already exists.
                    logger.info("The key pair '" + keyName + "' already exists.
Moving on...");
                } else {
                    logger.info("EC2 error occurred: Error message: {}, Error code
{}", ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
                    return;
                }
            } else {
                logger.info("An unexpected error occurred: " + (rt.getMessage()));
                return;
            }
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("2. List key pairs.");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<DescribeKeyPairsResponse> future =
ec2Actions.describeKeysAsync();
            DescribeKeyPairsResponse keyPairsResponse = future.join();
            keyPairsResponse.keyPairs().forEach(keyPair -> logger.info(
                "Found key pair with name {} and fingerprint {}",
                keyPair.keyName(),
                keyPair.keyFingerprint()));

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof Ec2Exception ec2Ex) {
                logger.info("EC2 error occurred: Error message: {}, Error code {}",
ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
                return;
            } else {
                logger.info("An unexpected error occurred: {}", (cause != null ?
cause.getMessage() : rt.getMessage()));
```

```
                return;
            }
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("3. Create a security group.");
        logger.info("""
            An AWS EC2 Security Group is a virtual firewall that controls the
            inbound and outbound traffic to an EC2 instance. It acts as a first
line
            of defense for your EC2 instances, allowing you to specify the rules
that
            govern the network traffic entering and leaving your instances.
           """);
        waitForInputToContinue(scanner);
        String groupId = "";
        try {
            CompletableFuture<String> future =
ec2Actions.createSecurityGroupAsync(groupName, groupDesc, vpcId, myIpAddress);
            future.join();
            logger.info("Created security group") ;

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof Ec2Exception ec2Ex) {
                if (ec2Ex.awsErrorDetails().errorMessage().contains("already
exists")) {
                    logger.info("The Security Group already exists. Moving on...");
                } else {
                    logger.error("An unexpected error occurred: {}",
ec2Ex.awsErrorDetails().errorMessage());
                    return;
                }
            } else {
                logger.error("An unexpected error occurred: {}",
cause.getMessage());
                return;
            }
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);
```

```
        logger.info(DASHES);
        logger.info("4. Display security group information for the new security
group.");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<String> future =
ec2Actions.describeSecurityGroupArnByNameAsync(groupName);
            groupId = future.join();
            logger.info("The security group Id is "+groupId);

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof Ec2Exception ec2Ex) {
                String errorCode = ec2Ex.awsErrorDetails().errorCode();
                if ("InvalidGroup.NotFound".equals(errorCode)) {
                    logger.info("Security group '{}' does not exist. Error Code:
{}", groupName, errorCode);
                } else {
                    logger.info("EC2 error occurred: Message {}, Error Code: {}",
ec2Ex.getMessage(), errorCode);
                }
            } else {
                logger.info("An unexpected error occurred: {}", cause.getMessage());
            }
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("5. Get a list of Amazon Linux 2 AMIs and select one with amzn2
in the name.");
        logger.info("""
            An Amazon EC2 AMI (Amazon Machine Image) is a pre-configured virtual
machine image that
            serves as a template for launching EC2 instances. It contains all the
necessary software and
            configurations required to run an application or operating system on an
EC2 instance.
            """);
        waitForInputToContinue(scanner);
        String instanceAMI="";
        try {
            CompletableFuture<GetParametersByPathResponse> future =
ec2Actions.getParaValuesAsync();
```

```
            GetParametersByPathResponse pathResponse = future.join();
            List<Parameter> parameterList = pathResponse.parameters();
            for (Parameter para : parameterList) {
                if (filterName(para.name())) {
                    instanceAMI = para.value();
                    break;
                }
            }
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof Ec2Exception ec2Ex) {
                logger.info("EC2 error occurred: Message {}, Error Code:{}",
ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
                return;
            } else {
                logger.info("An unexpected error occurred: {}", cause.getMessage());
                return;
            }
        }
        logger.info("The AMI value with amzn2 is: {}", instanceAMI);
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("6. Get the (Amazon Machine Image) AMI value from the amzn2
image.");
        logger.info("""
           An AMI value represents a specific version of a virtual machine (VM) or
server image.
           It uniquely identifies a particular version of an EC2 instance, including
its operating system,
           pre-installed software, and any custom configurations. This allows you to
consistently deploy the same
           VM image across your infrastructure.

            """);
        waitForInputToContinue(scanner);
        String amiValue;
        try {
            CompletableFuture<String> future =
ec2Actions.describeImageAsync(instanceAMI);
            amiValue = future.join();

        } catch (CompletionException ce) {
```

```
            Throwable cause = ce.getCause();
            if (cause instanceof Ec2Exception) {
                Ec2Exception ec2Ex = (Ec2Exception) cause;
                logger.info("EC2 error occurred: Message {}, Error Code:{}",
ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
                return;
            } else {
                logger.info("An unexpected error occurred: {}", cause.getMessage());
                return;
            }
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("7. Retrieves an instance type available in the current AWS
region.");
        waitForInputToContinue(scanner);
        String instanceType;
        try {
            CompletableFuture<String> future = ec2Actions.getInstanceTypesAsync();
            instanceType = future.join();
            if (!instanceType.isEmpty()) {
                logger.info("Found instance type: " + instanceType);
            } else {
                logger.info("Desired instance type not found.");
            }
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof Ec2Exception ec2Ex) {
                logger.info("EC2 error occurred: Message {}, Error Code:{}",
ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
                return;
            } else {
                logger.info("An unexpected error occurred: {}", cause.getMessage());
                return;
            }
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("8. Create an Amazon EC2 instance using the key pair, the
instance type, the security group, and the EC2 AMI value.");
```

```
        logger.info("Once the EC2 instance is created, it is placed into a running
state.");
        waitForInputToContinue(scanner);
        String newInstanceId;
        try {
            CompletableFuture<String> future =
ec2Actions.runInstanceAsync(instanceType, keyName, groupName, amiValue);
            newInstanceId = future.join();
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof Ec2Exception) {
                Ec2Exception ec2Ex = (Ec2Exception) cause;
                switch (ec2Ex.awsErrorDetails().errorCode()) {
                    case "InvalidParameterValue":
                        logger.info("EC2 error occurred: Message {}, Error Code:{}",
ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
                        break;
                    case "InsufficientInstanceCapacity":
                        // Handle insufficient instance capacity.
                        logger.info("Insufficient instance capacity: {}, {}",
ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
                        break;
                    case "InvalidGroup.NotFound":
                        // Handle security group not found.
                        logger.info("Security group not found: {},{}",
ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
                        break;
                    default:
                        logger.info("EC2 error occurred: {} (Code: {}",
ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
                        break;
                }
                return;
            } else {
                logger.info("An unexpected error occurred: {}", (cause != null ?
cause.getMessage() : rt.getMessage()));
                return;
            }
        }
        logger.info("The instance Id is " + newInstanceId);
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
```

```
        logger.info("9. Display information about the running instance. ");

        waitForInputToContinue(scanner);
        String publicIp;
        try {
            CompletableFuture<String> future =
ec2Actions.describeEC2InstancesAsync(newInstanceId);
            publicIp = future.join();
            logger.info("EC2 instance public IP {}", publicIp);
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof Ec2Exception ec2Ex) {
                logger.info("EC2 error occurred: Message {}, Error Code:{}",
ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
                return;
            } else {
                logger.info("An unexpected error occurred: {}", cause.getMessage());
                return;
            }

        }
        logger.info("You can SSH to the instance using this command:");
        logger.info("ssh -i " + fileName + " ec2-user@" + publicIp);
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("10. Stop the instance using a waiter (this may take a few
mins).");
        // Remove the 2nd one
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Void> future =
ec2Actions.stopInstanceAsync(newInstanceId);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof Ec2Exception ec2Ex) {
                logger.info("EC2 error occurred: Message {}, Error Code:{}",
ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
                return;
            } else {
                logger.info("An unexpected error occurred: {}", cause.getMessage());
```

```
                return;
            }
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("11. Start the instance using a waiter (this may take a few
mins).");
        try {
            CompletableFuture<Void> future =
ec2Actions.startInstanceAsync(newInstanceId);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof Ec2Exception ec2Ex) {
                // Handle EC2 exceptions.
                logger.info("EC2 error occurred: Message {}, Error Code:{}",
ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
                return;
            } else {
                logger.info("An unexpected error occurred: {}", cause.getMessage());
                return;
            }
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("12. Allocate an Elastic IP address and associate it with the
instance.");
        logger.info("""
            An Elastic IP address is a static public IP address that you can
associate with your EC2 instance.
            This allows you to have a fixed, predictable IP address that remains the
same even if your instance
            is stopped, terminated, or replaced.
            This is particularly useful for applications or services that need to be
accessed consistently from a
            known IP address.
```

```
            An EC2 Allocation ID (also known as a Reserved Instance Allocation
ID) is a unique identifier associated with a Reserved Instance (RI) that you have
purchased in AWS.

            When you purchase a Reserved Instance, AWS assigns a unique Allocation
ID to it.
            This Allocation ID is used to track and identify the specific RI you
have purchased,
            and it is important for managing and monitoring your Reserved Instances.

            """);

        waitForInputToContinue(scanner);
        String allocationId;
        try {
            CompletableFuture<String> future = ec2Actions.allocateAddressAsync();
            allocationId = future.join();
            logger.info("Successfully allocated address with ID: " +allocationId);
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof Ec2Exception ec2Ex) {
                logger.info("EC2 error occurred: Message {}, Error Code:{}",
ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
                return;
            } else {
                logger.info("An unexpected error occurred: {}", cause.getMessage());
                return;
            }
        }
        logger.info("The allocation Id value is " + allocationId);
        waitForInputToContinue(scanner);
        String associationId;
        try {
            CompletableFuture<String> future =
ec2Actions.associateAddressAsync(newInstanceId, allocationId);
            associationId = future.join();
            logger.info("Successfully associated address with ID: " +associationId);
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof Ec2Exception ec2Ex) {
                logger.info("EC2 error occurred: Message {}, Error Code:{}",
ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
                return;
            } else {
```

```
                logger.info("An unexpected error occurred: {}", cause.getMessage());
                return;
            }
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("13. Describe the instance again. Note that the public IP
address has changed");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<String> future =
ec2Actions.describeEC2InstancesAsync(newInstanceId);
            publicIp = future.join();
            logger.info("EC2 instance public IP: " + publicIp);
            logger.info("You can SSH to the instance using this command:");
            logger.info("ssh -i " + fileName + " ec2-user@" + publicIp);
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof Ec2Exception ec2Ex) {
                logger.info("EC2 error occurred: Message {}, Error Code:{}",
ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
                return;
            } else {
                logger.info("An unexpected error occurred: {}", cause.getMessage());
                return;
            }
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("14. Disassociate and release the Elastic IP address.");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<DisassociateAddressResponse> future =
ec2Actions.disassociateAddressAsync(associationId);
            future.join();
            logger.info("Address successfully disassociated.");
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof Ec2Exception ec2Ex) {
                // Handle EC2 exceptions.
```

```
                logger.info("EC2 error occurred: Message {}, Error Code:{}",
ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
                return;
            } else {
                logger.info("An unexpected error occurred: {}", cause.getMessage());
                return;
            }
        }
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<ReleaseAddressResponse> future =
ec2Actions.releaseEC2AddressAsync(allocationId);
            future.join(); // Wait for the operation to complete
            logger.info("Elastic IP address successfully released.");
        } catch (RuntimeException rte) {
            logger.info("An unexpected error occurred: {}", rte.getMessage());
            return;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("15. Terminate the instance and use a waiter (this may take a
few mins).");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Object> future =
ec2Actions.terminateEC2Async(newInstanceId);
            future.join();
            logger.info("EC2 instance successfully terminated.");
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof Ec2Exception ec2Ex) {
                // Handle EC2 exceptions.
                logger.info("EC2 error occurred: Message {}, Error Code:{}",
ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
                return;
            } else {
                logger.info("An unexpected error occurred: {}", cause.getMessage());
                return;
            }
        }
        logger.info(DASHES);
```

```
        logger.info(DASHES);
        logger.info("16. Delete the security group.");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Void> future =
ec2Actions.deleteEC2SecGroupAsync(groupId);
            future.join();
            logger.info("Security group successfully deleted.");
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof Ec2Exception ec2Ex) {
                logger.info("EC2 error occurred: Message {}, Error Code:{}",
ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
                return;
            } else {
                logger.info("An unexpected error occurred: {}", cause.getMessage());
                return;
            }
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("17. Delete the key.");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<DeleteKeyPairResponse> future =
ec2Actions.deleteKeysAsync(keyName);
            future.join();
            logger.info("Successfully deleted key pair named " + keyName);
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof Ec2Exception ec2Ex) {
                logger.info("EC2 error occurred: Message {}, Error Code:{}",
ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
                return;
            } else {
                logger.info("An unexpected error occurred: {}", cause.getMessage());
                return;
            }
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);
```

```
        logger.info(DASHES);
        logger.info("You successfully completed the Amazon EC2 scenario.");
        logger.info(DASHES);
    }
    public static boolean filterName(String name) {
        String[] parts = name.split("/");
        String myValue = parts[4];
        return myValue.contains("amzn2");
    }

    private static void waitForInputToContinue(Scanner scanner) {
        while (true) {
            logger.info("");
            logger.info("Enter 'c' followed by <ENTER> to continue:");
            String input = scanner.nextLine();

            if (input.trim().equalsIgnoreCase("c")) {
                logger.info("Continuing with the program...");
                logger.info("");
                break;
            } else {
                // Handle invalid input.
                logger.info("Invalid input. Please try again.");
            }
        }
    }
}
```

Definisci una classe che racchiuda le azioni. EC2

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.client.config.ClientOverrideConfiguration;
import software.amazon.awssdk.http.async.SdkAsyncHttpClient;
import software.amazon.awssdk.http.nio.netty.NettyNioAsyncHttpClient;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2AsyncClient;
import software.amazon.awssdk.services.ec2.model.AllocateAddressRequest;
import software.amazon.awssdk.services.ec2.model.AllocateAddressResponse;
import software.amazon.awssdk.services.ec2.model.AssociateAddressRequest;
import software.amazon.awssdk.services.ec2.model.AssociateAddressResponse;
```

```
import
 software.amazon.awssdk.services.ec2.model.AuthorizeSecurityGroupIngressRequest;
import software.amazon.awssdk.services.ec2.model.CreateKeyPairRequest;
import software.amazon.awssdk.services.ec2.model.CreateKeyPairResponse;
import software.amazon.awssdk.services.ec2.model.CreateSecurityGroupRequest;
import software.amazon.awssdk.services.ec2.model.DeleteKeyPairRequest;
import software.amazon.awssdk.services.ec2.model.DeleteKeyPairResponse;
import software.amazon.awssdk.services.ec2.model.DeleteSecurityGroupRequest;
import software.amazon.awssdk.services.ec2.model.DeleteSecurityGroupResponse;
import software.amazon.awssdk.services.ec2.model.DescribeImagesRequest;
import software.amazon.awssdk.services.ec2.model.DescribeInstanceTypesRequest;
import software.amazon.awssdk.services.ec2.model.DescribeInstanceTypesResponse;
import software.amazon.awssdk.services.ec2.model.DescribeInstancesRequest;
import software.amazon.awssdk.services.ec2.model.DescribeKeyPairsResponse;
import software.amazon.awssdk.services.ec2.model.DescribeSecurityGroupsRequest;
import software.amazon.awssdk.services.ec2.model.DescribeSecurityGroupsResponse;
import software.amazon.awssdk.services.ec2.model.DescribeVpcsRequest;
import software.amazon.awssdk.services.ec2.model.DisassociateAddressRequest;
import software.amazon.awssdk.services.ec2.model.DisassociateAddressResponse;
import software.amazon.awssdk.services.ec2.model.DomainType;
import software.amazon.awssdk.services.ec2.model.Ec2Exception;
import software.amazon.awssdk.services.ec2.model.Filter;
import software.amazon.awssdk.services.ec2.model.InstanceTypeInfo;
import software.amazon.awssdk.services.ec2.model.IpPermission;
import software.amazon.awssdk.services.ec2.model.IpRange;
import software.amazon.awssdk.services.ec2.model.ReleaseAddressRequest;
import software.amazon.awssdk.services.ec2.model.ReleaseAddressResponse;
import software.amazon.awssdk.services.ec2.model.RunInstancesRequest;
import software.amazon.awssdk.services.ec2.model.RunInstancesResponse;
import software.amazon.awssdk.services.ec2.model.StopInstancesRequest;
import software.amazon.awssdk.services.ec2.model.StartInstancesRequest;
import software.amazon.awssdk.services.ec2.model.TerminateInstancesRequest;
import software.amazon.awssdk.services.ec2.model.Vpc;
import software.amazon.awssdk.services.ec2.paginators.DescribeImagesPublisher;
import software.amazon.awssdk.services.ec2.paginators.DescribeInstancesPublisher;
import
 software.amazon.awssdk.services.ec2.paginators.DescribeSecurityGroupsPublisher;
import software.amazon.awssdk.services.ec2.paginators.DescribeVpcsPublisher;
import software.amazon.awssdk.services.ec2.waiters.Ec2AsyncWaiter;
import software.amazon.awssdk.services.ssm.SsmAsyncClient;
import software.amazon.awssdk.services.ssm.model.GetParametersByPathRequest;
import software.amazon.awssdk.services.ssm.model.GetParametersByPathResponse;
import software.amazon.awssdk.services.ec2.model.TerminateInstancesResponse;
import java.io.BufferedWriter;
```

```
import java.io.FileWriter;
import java.io.IOException;
import java.time.Duration;
import java.util.List;
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.CompletionException;
import java.util.concurrent.atomic.AtomicReference;

public class EC2Actions {
    private static final Logger logger = LoggerFactory.getLogger(EC2Actions.class);
    private static Ec2AsyncClient ec2AsyncClient;

    /**
     * Retrieves an asynchronous Amazon Elastic Container Registry (ECR) client.
     *
     * @return the configured ECR asynchronous client.
     */
    private static Ec2AsyncClient getAsyncClient() {
        if (ec2AsyncClient == null) {
            /*
            The `NettyNioAsyncHttpClient` class is part of the AWS SDK for Java,
 version 2,
            and it is designed to provide a high-performance, asynchronous HTTP
 client for interacting with AWS services.
             It uses the Netty framework to handle the underlying network
 communication and the Java NIO API to
             provide a non-blocking, event-driven approach to HTTP requests and
 responses.
             */
            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                .maxConcurrency(50)  // Adjust as needed.
                .connectionTimeout(Duration.ofSeconds(60))  // Set the connection
 timeout.
                .readTimeout(Duration.ofSeconds(60))  // Set the read timeout.
                .writeTimeout(Duration.ofSeconds(60))  // Set the write timeout.
                .build();

            ClientOverrideConfiguration overrideConfig =
 ClientOverrideConfiguration.builder()
               .apiCallTimeout(Duration.ofMinutes(2))  // Set the overall API call
 timeout.
                .apiCallAttemptTimeout(Duration.ofSeconds(90))  // Set the
 individual call attempt timeout.
                .build();
```

```
            ec2AsyncClient = Ec2AsyncClient.builder()
                .region(Region.US_EAST_1)
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return ec2AsyncClient;
    }

    /**
     * Deletes a key pair asynchronously.
     *
     * @param keyPair the name of the key pair to delete
     * @return a {@link CompletableFuture} that represents the result of the
asynchronous operation.
     *         The {@link CompletableFuture} will complete with a {@link
DeleteKeyPairResponse} object
     *         that provides the result of the key pair deletion operation.
     */
    public CompletableFuture<DeleteKeyPairResponse> deleteKeysAsync(String keyPair)
{
        DeleteKeyPairRequest request = DeleteKeyPairRequest.builder()
            .keyName(keyPair)
            .build();

        // Initiate the asynchronous request to delete the key pair.
        CompletableFuture<DeleteKeyPairResponse> response =
getAsyncClient().deleteKeyPair(request);
        return response.whenComplete((resp, ex) -> {
            if (ex != null) {
                throw new RuntimeException("Failed to delete key pair: " + keyPair,
ex);
            } else if (resp == null) {
                throw new RuntimeException("No response received for deleting key
pair: " + keyPair);
            }
        });
    }

    /**
     * Deletes an EC2 security group asynchronously.
     *
     * @param groupId the ID of the security group to delete
```

```
     * @return a CompletableFuture that completes when the security group is deleted
     */
    public CompletableFuture<Void> deleteEC2SecGroupAsync(String groupId) {
        DeleteSecurityGroupRequest request = DeleteSecurityGroupRequest.builder()
            .groupId(groupId)
            .build();

        CompletableFuture<DeleteSecurityGroupResponse> response =
getAsyncClient().deleteSecurityGroup(request);
        return response.whenComplete((resp, ex) -> {
            if (ex != null) {
                throw new RuntimeException("Failed to delete security group with Id
" + groupId, ex);
            } else if (resp == null) {
                throw new RuntimeException("No response received for deleting
security group with Id " + groupId);
            }
        }).thenApply(resp -> null);
    }

    /**
     * Terminates an EC2 instance asynchronously and waits for it to reach the
terminated state.
     *
     * @param instanceId the ID of the EC2 instance to terminate
     * @return a {@link CompletableFuture} that completes when the instance has been
terminated
     * @throws RuntimeException if there is no response from the AWS SDK or if there
is a failure during the termination process
     */
    public CompletableFuture<Object> terminateEC2Async(String instanceId) {
        TerminateInstancesRequest terminateRequest =
TerminateInstancesRequest.builder()
            .instanceIds(instanceId)
            .build();

        CompletableFuture<TerminateInstancesResponse> responseFuture =
getAsyncClient().terminateInstances(terminateRequest);
        return responseFuture.thenCompose(terminateResponse -> {
            if (terminateResponse == null) {
                throw new RuntimeException("No response received for terminating
instance " + instanceId);
            }
```

```
            System.out.println("Going to terminate an EC2 instance and use a waiter
to wait for it to be in terminated state");
            return getAsyncClient().waiter()
                .waitUntilInstanceTerminated(r -> r.instanceIds(instanceId))
                .thenApply(waiterResponse -> null);
        }).exceptionally(throwable -> {
            // Handle any exceptions that occurred during the async call
            throw new RuntimeException("Failed to terminate EC2 instance: " +
throwable.getMessage(), throwable);
        });
    }

    /**
     * Releases an Elastic IP address asynchronously.
     *
     * @param allocId the allocation ID of the Elastic IP address to be released
     * @return a {@link CompletableFuture} representing the asynchronous operation
of releasing the Elastic IP address
     */
    public CompletableFuture<ReleaseAddressResponse> releaseEC2AddressAsync(String
allocId) {
        ReleaseAddressRequest request = ReleaseAddressRequest.builder()
            .allocationId(allocId)
            .build();

        CompletableFuture<ReleaseAddressResponse> response =
getAsyncClient().releaseAddress(request);
        response.whenComplete((resp, ex) -> {
            if (ex != null) {
                throw new RuntimeException("Failed to release Elastic IP address",
ex);
            }
        });

        return response;
    }

    /**
     * Disassociates an Elastic IP address from an instance asynchronously.
     *
     * @param associationId The ID of the association you want to disassociate.
     * @return a {@link CompletableFuture} representing the asynchronous operation
of disassociating the address. The
```

```
    *         {@link CompletableFuture} will complete with a {@link
DisassociateAddressResponse} when the operation is
    *         finished.
    * @throws RuntimeException if the disassociation of the address fails.
    */
   public CompletableFuture<DisassociateAddressResponse>
disassociateAddressAsync(String associationId) {
       Ec2AsyncClient ec2 = getAsyncClient();
       DisassociateAddressRequest addressRequest =
DisassociateAddressRequest.builder()
           .associationId(associationId)
           .build();

       // Disassociate the address asynchronously.
       CompletableFuture<DisassociateAddressResponse> response =
ec2.disassociateAddress(addressRequest);
       response.whenComplete((resp, ex) -> {
           if (ex != null) {
              throw new RuntimeException("Failed to disassociate address", ex);
           }
       });

       return response;
   }

   /**
    * Associates an Elastic IP address with an EC2 instance asynchronously.
    *
    * @param instanceId    the ID of the EC2 instance to associate the Elastic IP
address with
    * @param allocationId  the allocation ID of the Elastic IP address to associate
    * @return a {@link CompletableFuture} that completes with the association ID
when the operation is successful,
    *         or throws a {@link RuntimeException} if the operation fails
    */
   public CompletableFuture<String> associateAddressAsync(String instanceId, String
allocationId) {
       AssociateAddressRequest associateRequest = AssociateAddressRequest.builder()
           .instanceId(instanceId)
           .allocationId(allocationId)
           .build();

       CompletableFuture<AssociateAddressResponse> responseFuture =
getAsyncClient().associateAddress(associateRequest);
```

```
        return responseFuture.thenApply(response -> {
            if (response.associationId() != null) {
                return response.associationId();
            } else {
                throw new RuntimeException("Association ID is null after associating
address.");
            }
        }).whenComplete((result, ex) -> {
            if (ex != null) {
                throw new RuntimeException("Failed to associate address", ex);
            }
        });
    }

    /**
     * Allocates an Elastic IP address asynchronously in the VPC domain.
     *
     * @return a {@link CompletableFuture} containing the allocation ID of the
allocated Elastic IP address
     */
    public CompletableFuture<String> allocateAddressAsync() {
        AllocateAddressRequest allocateRequest = AllocateAddressRequest.builder()
            .domain(DomainType.VPC)
            .build();

        CompletableFuture<AllocateAddressResponse> responseFuture =
getAsyncClient().allocateAddress(allocateRequest);
        return
responseFuture.thenApply(AllocateAddressResponse::allocationId).whenComplete((result,
ex) -> {
            if (ex != null) {
                throw new RuntimeException("Failed to allocate address", ex);
            }
        });
    }

    /**
     * Asynchronously describes the state of an EC2 instance.
     * The paginator helps you iterate over multiple pages of results.
     *
     * @param newInstanceId the ID of the EC2 instance to describe
     * @return a {@link CompletableFuture} that, when completed, contains a string
describing the state of the EC2 instance
     */
```

```
   public CompletableFuture<String> describeEC2InstancesAsync(String newInstanceId)
{
       DescribeInstancesRequest request = DescribeInstancesRequest.builder()
           .instanceIds(newInstanceId)
           .build();

       DescribeInstancesPublisher paginator =
getAsyncClient().describeInstancesPaginator(request);
       AtomicReference<String> publicIpAddressRef = new AtomicReference<>();
       return paginator.subscribe(response -> {
           response.reservations().stream()
               .flatMap(reservation -> reservation.instances().stream())
               .filter(instance -> instance.instanceId().equals(newInstanceId))
               .findFirst()
               .ifPresent(instance ->
publicIpAddressRef.set(instance.publicIpAddress()));
       }).thenApply(v -> {
           String publicIpAddress = publicIpAddressRef.get();
           if (publicIpAddress == null) {
               throw new RuntimeException("Instance with ID " + newInstanceId + "
not found.");
           }
           return publicIpAddress;
       }).exceptionally(ex -> {
           logger.info("Failed to describe instances: " + ex.getMessage());
           throw new RuntimeException("Failed to describe instances", ex);
       });
   }

   /**
    * Runs an EC2 instance asynchronously.
    *
    * @param instanceType The instance type to use for the EC2 instance.
    * @param keyName The name of the key pair to associate with the EC2 instance.
    * @param groupName The name of the security group to associate with the EC2
instance.
    * @param amiId The ID of the Amazon Machine Image (AMI) to use for the EC2
instance.
    * @return A {@link CompletableFuture} that completes with the ID of the started
EC2 instance.
    * @throws RuntimeException If there is an error running the EC2 instance.
    */
   public CompletableFuture<String> runInstanceAsync(String instanceType, String
keyName, String groupName, String amiId) {
```

```
        RunInstancesRequest runRequest = RunInstancesRequest.builder()
            .instanceType(instanceType)
            .keyName(keyName)
            .securityGroups(groupName)
            .maxCount(1)
            .minCount(1)
            .imageId(amiId)
            .build();

        CompletableFuture<RunInstancesResponse> responseFuture =
getAsyncClient().runInstances(runRequest);
        return responseFuture.thenCompose(response -> {
            String instanceIdVal = response.instances().get(0).instanceId();
            System.out.println("Going to start an EC2 instance and use a waiter to
wait for it to be in running state");
            return getAsyncClient().waiter()
                .waitUntilInstanceExists(r -> r.instanceIds(instanceIdVal))
                .thenCompose(waitResponse -> getAsyncClient().waiter()
                    .waitUntilInstanceRunning(r -> r.instanceIds(instanceIdVal))
                    .thenApply(runningResponse -> instanceIdVal));
        }).exceptionally(throwable -> {
            // Handle any exceptions that occurred during the async call
            throw new RuntimeException("Failed to run EC2 instance: " +
throwable.getMessage(), throwable);
        });
    }

    /**
     * Asynchronously retrieves the instance types available in the current AWS
region.
     * <p>
     * This method uses the AWS SDK's asynchronous API to fetch the available
instance types
     * and then processes the response. It logs the memory information, network
information,
     * and instance type for each instance type returned. Additionally, it returns a
     * {@link CompletableFuture} that resolves to the instance type string for the
"t2.2xlarge"
     * instance type, if it is found in the response. If the "t2.2xlarge" instance
type is not
     * found, an empty string is returned.
     * </p>
     *
```

```
    * @return a {@link CompletableFuture} that resolves to the instance type string
for the
    * "t2.2xlarge" instance type, or an empty string if the instance type is not
found
    */
   public CompletableFuture<String> getInstanceTypesAsync() {
       DescribeInstanceTypesRequest typesRequest =
DescribeInstanceTypesRequest.builder()
           .maxResults(10)
           .build();

       CompletableFuture<DescribeInstanceTypesResponse> response =
getAsyncClient().describeInstanceTypes(typesRequest);
       response.whenComplete((resp, ex) -> {
           if (resp != null) {
               List<InstanceTypeInfo> instanceTypes = resp.instanceTypes();
               for (InstanceTypeInfo type : instanceTypes) {
                   logger.info("The memory information of this type is " +
type.memoryInfo().sizeInMiB());
                   logger.info("Network information is " +
type.networkInfo().toString());
                   logger.info("Instance type is " +
type.instanceType().toString());
               }
           } else {
               throw (RuntimeException) ex;
           }
       });

       return response.thenApply(resp -> {
           for (InstanceTypeInfo type : resp.instanceTypes()) {
               String instanceType = type.instanceType().toString();
               if (instanceType.equals("t2.2xlarge")) {
                   return instanceType;
               }
           }
           return "";
       });
   }

   /**
    * Asynchronously describes an AWS EC2 image with the specified image ID.
    *
    * @param imageId the ID of the image to be described
```

```
    * @return a {@link CompletableFuture} that, when completed, contains the ID of
the described image
    * @throws RuntimeException if no images are found with the provided image ID,
or if an error occurs during the AWS API call
    */
   public CompletableFuture<String> describeImageAsync(String imageId) {
       DescribeImagesRequest imagesRequest = DescribeImagesRequest.builder()
           .imageIds(imageId)
           .build();

       AtomicReference<String> imageIdRef = new AtomicReference<>();
       DescribeImagesPublisher paginator =
getAsyncClient().describeImagesPaginator(imagesRequest);
       return paginator.subscribe(response -> {
           response.images().stream()
               .filter(image -> image.imageId().equals(imageId))
               .findFirst()
               .ifPresent(image -> {
                   logger.info("The description of the image is " +
image.description());
                   logger.info("The name of the image is " + image.name());
                   imageIdRef.set(image.imageId());
               });
       }).thenApply(v -> {
           String id = imageIdRef.get();
           if (id == null) {
               throw new RuntimeException("No images found with the provided image
ID.");
           }
           return id;
       }).exceptionally(ex -> {
           logger.info("Failed to describe image: " + ex.getMessage());
           throw new RuntimeException("Failed to describe image", ex);
       });
   }

   /**
    * Retrieves the parameter values asynchronously using the AWS Systems Manager
(SSM) API.
    *
    * @return a {@link CompletableFuture} that holds the response from the SSM API
call to get parameters by path
    */
   public CompletableFuture<GetParametersByPathResponse> getParaValuesAsync() {
```

```
        SsmAsyncClient ssmClient = SsmAsyncClient.builder()
            .region(Region.US_EAST_1)
            .build();

        GetParametersByPathRequest parameterRequest =
GetParametersByPathRequest.builder()
            .path("/aws/service/ami-amazon-linux-latest")
            .build();

        // Create a CompletableFuture to hold the final result.
        CompletableFuture<GetParametersByPathResponse> responseFuture = new
CompletableFuture<>();
        ssmClient.getParametersByPath(parameterRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    responseFuture.completeExceptionally(new
RuntimeException("Failed to get parameters by path", exception));
                } else {
                    responseFuture.complete(response);
                }
            });

        return responseFuture;
    }


    /**
     * Asynchronously describes the security groups for the specified group ID.
     *
     * @param groupName the name of the security group to describe
     * @return a {@link CompletableFuture} that represents the asynchronous
operation
     *         of describing the security groups. The future will complete with a
     *         {@link DescribeSecurityGroupsResponse} object that contains the
     *         security group information.
     */
    public CompletableFuture<String> describeSecurityGroupArnByNameAsync(String
groupName) {
        DescribeSecurityGroupsRequest request =
DescribeSecurityGroupsRequest.builder()
            .groupNames(groupName)
            .build();
```

```
        DescribeSecurityGroupsPublisher paginator =
getAsyncClient().describeSecurityGroupsPaginator(request);
        AtomicReference<String> groupIdRef = new AtomicReference<>();
        return paginator.subscribe(response -> {
            response.securityGroups().stream()
                .filter(securityGroup ->
securityGroup.groupName().equals(groupName))
                .findFirst()
                .ifPresent(securityGroup ->
groupIdRef.set(securityGroup.groupId()));
        }).thenApply(v -> {
            String groupId = groupIdRef.get();
            if (groupId == null) {
                throw new RuntimeException("No security group found with the name: "
+ groupName);
            }
            return groupId;
        }).exceptionally(ex -> {
            logger.info("Failed to describe security group: " + ex.getMessage());
            throw new RuntimeException("Failed to describe security group", ex);
        });
    }

    /**
     * Creates a new security group asynchronously with the specified group name,
description, and VPC ID. It also
     * authorizes inbound traffic on ports 80 and 22 from the specified IP address.
     *
     * @param groupName    the name of the security group to create
     * @param groupDesc    the description of the security group
     * @param vpcId        the ID of the VPC in which to create the security group
     * @param myIpAddress  the IP address from which to allow inbound traffic (e.g.,
"192.168.1.1/0" to allow traffic from
     *                     any IP address in the 192.168.1.0/24 subnet)
     * @return a CompletableFuture that, when completed, returns the ID of the
created security group
     * @throws RuntimeException if there was a failure creating the security group
or authorizing the inbound traffic
     */
    public CompletableFuture<String> createSecurityGroupAsync(String groupName,
String groupDesc, String vpcId, String myIpAddress) {
        CreateSecurityGroupRequest createRequest =
CreateSecurityGroupRequest.builder()
            .groupName(groupName)
```

```
            .description(groupDesc)
            .vpcId(vpcId)
            .build();

        return getAsyncClient().createSecurityGroup(createRequest)
            .thenCompose(createResponse -> {
                String groupId = createResponse.groupId();
                IpRange ipRange = IpRange.builder()
                    .cidrIp(myIpAddress + "/32")
                    .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();

                return getAsyncClient().authorizeSecurityGroupIngress(authRequest)
                    .thenApply(authResponse -> groupId);
            })
            .whenComplete((result, exception) -> {
                if (exception != null) {
                    if (exception instanceof CompletionException &&
exception.getCause() instanceof Ec2Exception) {
                        throw (Ec2Exception) exception.getCause();
                    } else {
                        throw new RuntimeException("Failed to create security group:
" + exception.getMessage(), exception);
                    }
                }
```

```
            });
    }

    /**
     * Asynchronously describes the key pairs associated with the current AWS
account.
     *
     * @return a {@link CompletableFuture} containing the {@link
DescribeKeyPairsResponse} object, which provides
     * information about the key pairs.
     */
    public CompletableFuture<DescribeKeyPairsResponse> describeKeysAsync() {
        CompletableFuture<DescribeKeyPairsResponse> responseFuture =
getAsyncClient().describeKeyPairs();
        responseFuture.whenComplete((response, exception) -> {
            if (exception != null) {
              throw new RuntimeException("Failed to describe key pairs: " +
exception.getMessage(), exception);
            }
        });

        return responseFuture;
    }

    /**
     * Creates a new key pair asynchronously.
     *
     * @param keyName the name of the key pair to create
     * @param fileName the name of the file to write the key material to
     * @return a {@link CompletableFuture} that represents the asynchronous
operation
     *         of creating the key pair and writing the key material to a file
     */
    public CompletableFuture<CreateKeyPairResponse> createKeyPairAsync(String
keyName, String fileName) {
        CreateKeyPairRequest request = CreateKeyPairRequest.builder()
            .keyName(keyName)
            .build();

        CompletableFuture<CreateKeyPairResponse> responseFuture =
getAsyncClient().createKeyPair(request);
        responseFuture.whenComplete((response, exception) -> {
            if (response != null) {
                try {
```

```
                    BufferedWriter writer = new BufferedWriter(new
FileWriter(fileName));
                    writer.write(response.keyMaterial());
                    writer.close();
                } catch (IOException e) {
                    throw new RuntimeException("Failed to write key material to
file: " + e.getMessage(), e);
                }
            } else {
                throw new RuntimeException("Failed to create key pair: " +
exception.getMessage(), exception);
            }
        });

        return responseFuture;
    }

    /**
     * Describes the first default VPC asynchronously and using a paginator.
     *
     * @return a {@link CompletableFuture} that, when completed, contains the first
default VPC found.\
     */
    public CompletableFuture<Vpc> describeFirstEC2VpcAsync() {
        Filter myFilter = Filter.builder()
            .name("is-default")
            .values("true")
            .build();

        DescribeVpcsRequest request = DescribeVpcsRequest.builder()
            .filters(myFilter)
            .build();

        DescribeVpcsPublisher paginator =
getAsyncClient().describeVpcsPaginator(request);
        AtomicReference<Vpc> vpcRef = new AtomicReference<>();
        return paginator.subscribe(response -> {
            response.vpcs().stream()
                .findFirst()
                .ifPresent(vpcRef::set);
        }).thenApply(v -> {
            Vpc vpc = vpcRef.get();
            if (vpc == null) {
                throw new RuntimeException("Default VPC not found");
```

```
            }
            return vpc;
        }).exceptionally(ex -> {
            logger.info("Failed to describe VPCs: " + ex.getMessage());
            throw new RuntimeException("Failed to describe VPCs", ex);
        });
    }

    /**
     * Stops the EC2 instance with the specified ID asynchronously and waits for the
instance to stop.
     *
     * @param instanceId the ID of the EC2 instance to stop
     * @return a {@link CompletableFuture} that completes when the instance has been
stopped, or exceptionally if an error occurs
     */
    public CompletableFuture<Void> stopInstanceAsync(String instanceId) {
        StopInstancesRequest stopRequest = StopInstancesRequest.builder()
            .instanceIds(instanceId)
            .build();

        DescribeInstancesRequest describeRequest =
DescribeInstancesRequest.builder()
            .instanceIds(instanceId)
            .build();

        Ec2AsyncWaiter ec2Waiter = Ec2AsyncWaiter.builder()
            .client(getAsyncClient())
            .build();

        CompletableFuture<Void> resultFuture = new CompletableFuture<>();
        logger.info("Stopping instance " + instanceId + " and waiting for it to
stop.");
        getAsyncClient().stopInstances(stopRequest)
            .thenCompose(response -> {
                if (response.stoppingInstances().isEmpty()) {
                    return CompletableFuture.failedFuture(new RuntimeException("No
instances were stopped. Please check the instance ID: " + instanceId));
                }
                return ec2Waiter.waitUntilInstanceStopped(describeRequest);
            })
            .thenAccept(waiterResponse -> {
                logger.info("Successfully stopped instance " + instanceId);
                resultFuture.complete(null);
```

```
            })
            .exceptionally(throwable -> {
                logger.error("Failed to stop instance " + instanceId + ": " +
throwable.getMessage(), throwable);
                resultFuture.completeExceptionally(new RuntimeException("Failed to
stop instance: " + throwable.getMessage(), throwable));
                return null;
            });

        return resultFuture;
    }

    /**
     * Starts an Amazon EC2 instance asynchronously and waits until it is in the
"running" state.
     *
     * @param instanceId the ID of the instance to start
     * @return a {@link CompletableFuture} that completes when the instance has been
started and is in the "running" state, or exceptionally if an error occurs
     */
    public CompletableFuture<Void> startInstanceAsync(String instanceId) {
        StartInstancesRequest startRequest = StartInstancesRequest.builder()
            .instanceIds(instanceId)
            .build();

        Ec2AsyncWaiter ec2Waiter = Ec2AsyncWaiter.builder()
            .client(getAsyncClient())
            .build();

        DescribeInstancesRequest describeRequest =
DescribeInstancesRequest.builder()
            .instanceIds(instanceId)
            .build();

        logger.info("Starting instance " + instanceId + " and waiting for it to
run.");
        CompletableFuture<Void> resultFuture = new CompletableFuture<>();
        return getAsyncClient().startInstances(startRequest)
            .thenCompose(response ->
                ec2Waiter.waitUntilInstanceRunning(describeRequest)
            )
            .thenAccept(waiterResponse -> {
                logger.info("Successfully started instance " + instanceId);
                resultFuture.complete(null);
```

```
            })
            .exceptionally(throwable -> {
                resultFuture.completeExceptionally(new RuntimeException("Failed to
 start instance: " + throwable.getMessage(), throwable));
                return null;
            });
    }

}
```

- 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

Azioni

`AllocateAddress`

Il seguente esempio di codice mostra come usare. `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.

```
   /**
    * Allocates an Elastic IP address asynchronously in the VPC domain.
    *
    * @return a {@link CompletableFuture} containing the allocation ID of the
allocated Elastic IP address
    */
   public CompletableFuture<String> allocateAddressAsync() {
       AllocateAddressRequest allocateRequest = AllocateAddressRequest.builder()
           .domain(DomainType.VPC)
           .build();

       CompletableFuture<AllocateAddressResponse> responseFuture =
getAsyncClient().allocateAddress(allocateRequest);
       return
responseFuture.thenApply(AllocateAddressResponse::allocationId).whenComplete((result,
ex) -> {
           if (ex != null) {
               throw new RuntimeException("Failed to allocate address", ex);
           }
       });
   }
```

- Per i dettagli sull'API, consulta la AllocateAddresssezione 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.

```
   /**
    * Associates an Elastic IP address with an EC2 instance asynchronously.
    *
    * @param instanceId    the ID of the EC2 instance to associate the Elastic IP
address with
    * @param allocationId  the allocation ID of the Elastic IP address to associate
    * @return a {@link CompletableFuture} that completes with the association ID
when the operation is successful,
    *         or throws a {@link RuntimeException} if the operation fails
    */
   public CompletableFuture<String> associateAddressAsync(String instanceId, String
allocationId) {
       AssociateAddressRequest associateRequest = AssociateAddressRequest.builder()
           .instanceId(instanceId)
           .allocationId(allocationId)
           .build();

       CompletableFuture<AssociateAddressResponse> responseFuture =
getAsyncClient().associateAddress(associateRequest);
       return responseFuture.thenApply(response -> {
           if (response.associationId() != null) {
               return response.associationId();
           } else {
               throw new RuntimeException("Association ID is null after associating
address.");
           }
       }).whenComplete((result, ex) -> {
           if (ex != null) {
               throw new RuntimeException("Failed to associate address", ex);
           }
       });
```

```
    }
```

- Per i dettagli sull'API, consulta la AssociateAddresssezione 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.

```
   /**
    * Creates a new security group asynchronously with the specified group name,
description, and VPC ID. It also
    * authorizes inbound traffic on ports 80 and 22 from the specified IP address.
    *
    * @param groupName    the name of the security group to create
    * @param groupDesc    the description of the security group
    * @param vpcId        the ID of the VPC in which to create the security group
    * @param myIpAddress  the IP address from which to allow inbound traffic (e.g.,
"192.168.1.1/0" to allow traffic from
    *                     any IP address in the 192.168.1.0/24 subnet)
    * @return a CompletableFuture that, when completed, returns the ID of the
created security group
    * @throws RuntimeException if there was a failure creating the security group
or authorizing the inbound traffic
    */
   public CompletableFuture<String> createSecurityGroupAsync(String groupName,
String groupDesc, String vpcId, String myIpAddress) {
       CreateSecurityGroupRequest createRequest =
CreateSecurityGroupRequest.builder()
           .groupName(groupName)
           .description(groupDesc)
           .vpcId(vpcId)
```

```
            .build();

        return getAsyncClient().createSecurityGroup(createRequest)
            .thenCompose(createResponse -> {
                String groupId = createResponse.groupId();
                IpRange ipRange = IpRange.builder()
                    .cidrIp(myIpAddress + "/32")
                    .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();

                return getAsyncClient().authorizeSecurityGroupIngress(authRequest)
                    .thenApply(authResponse -> groupId);
            })
            .whenComplete((result, exception) -> {
                if (exception != null) {
                    if (exception instanceof CompletionException &&
exception.getCause() instanceof Ec2Exception) {
                        throw (Ec2Exception) exception.getCause();
                    } else {
                        throw new RuntimeException("Failed to create security group:
" + exception.getMessage(), exception);
                    }
                }
            });
    }
```

- Per i dettagli sull'API, consulta la AuthorizeSecurityGroupIngresssezione 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.

```
   /**
    * Creates a new key pair asynchronously.
    *
    * @param keyName the name of the key pair to create
    * @param fileName the name of the file to write the key material to
    * @return a {@link CompletableFuture} that represents the asynchronous
operation
    *         of creating the key pair and writing the key material to a file
    */
   public CompletableFuture<CreateKeyPairResponse> createKeyPairAsync(String
keyName, String fileName) {
       CreateKeyPairRequest request = CreateKeyPairRequest.builder()
           .keyName(keyName)
           .build();

       CompletableFuture<CreateKeyPairResponse> responseFuture =
getAsyncClient().createKeyPair(request);
       responseFuture.whenComplete((response, exception) -> {
           if (response != null) {
               try {
                   BufferedWriter writer = new BufferedWriter(new
FileWriter(fileName));
                   writer.write(response.keyMaterial());
                   writer.close();
```

```
                } catch (IOException e) {
                    throw new RuntimeException("Failed to write key material to
file: " + e.getMessage(), e);
                }
            } else {
                throw new RuntimeException("Failed to create key pair: " +
exception.getMessage(), exception);
            }
        });

        return responseFuture;
    }
```

- Per i dettagli sull'API, consulta la CreateKeyPairsezione 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.

```
    /**
     * Creates a new security group asynchronously with the specified group name,
description, and VPC ID. It also
     * authorizes inbound traffic on ports 80 and 22 from the specified IP address.
     *
     * @param groupName    the name of the security group to create
     * @param groupDesc    the description of the security group
     * @param vpcId        the ID of the VPC in which to create the security group
     * @param myIpAddress  the IP address from which to allow inbound traffic (e.g.,
"192.168.1.1/0" to allow traffic from
     *                     any IP address in the 192.168.1.0/24 subnet)
     * @return a CompletableFuture that, when completed, returns the ID of the
created security group
```

```
     * @throws RuntimeException if there was a failure creating the security group
or authorizing the inbound traffic
     */
    public CompletableFuture<String> createSecurityGroupAsync(String groupName,
String groupDesc, String vpcId, String myIpAddress) {
        CreateSecurityGroupRequest createRequest =
CreateSecurityGroupRequest.builder()
            .groupName(groupName)
            .description(groupDesc)
            .vpcId(vpcId)
            .build();

        return getAsyncClient().createSecurityGroup(createRequest)
            .thenCompose(createResponse -> {
                String groupId = createResponse.groupId();
                IpRange ipRange = IpRange.builder()
                    .cidrIp(myIpAddress + "/32")
                    .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();

                return getAsyncClient().authorizeSecurityGroupIngress(authRequest)
                    .thenApply(authResponse -> groupId);
            })
            .whenComplete((result, exception) -> {
                if (exception != null) {
```

```
                    if (exception instanceof CompletionException &&
exception.getCause() instanceof Ec2Exception) {
                        throw (Ec2Exception) exception.getCause();
                    } else {
                        throw new RuntimeException("Failed to create security group:
" + exception.getMessage(), exception);
                    }
                }
            });
    }
```

- Per i dettagli sull'API, consulta la CreateSecurityGroupsezione 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.

```
    /**
     * Deletes a key pair asynchronously.
     *
     * @param keyPair the name of the key pair to delete
     * @return a {@link CompletableFuture} that represents the result of the
asynchronous operation.
     *         The {@link CompletableFuture} will complete with a {@link
DeleteKeyPairResponse} object
     *         that provides the result of the key pair deletion operation.
     */
    public CompletableFuture<DeleteKeyPairResponse> deleteKeysAsync(String keyPair)
{
        DeleteKeyPairRequest request = DeleteKeyPairRequest.builder()
            .keyName(keyPair)
```

```
            .build();

        // Initiate the asynchronous request to delete the key pair.
        CompletableFuture<DeleteKeyPairResponse> response =
getAsyncClient().deleteKeyPair(request);
        return response.whenComplete((resp, ex) -> {
            if (ex != null) {
                throw new RuntimeException("Failed to delete key pair: " + keyPair,
ex);
            } else if (resp == null) {
                throw new RuntimeException("No response received for deleting key
pair: " + keyPair);
            }
        });
    }
```

- Per i dettagli sull'API, consulta la DeleteKeyPairsezione 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.

```
    /**
     * Deletes an EC2 security group asynchronously.
     *
     * @param groupId the ID of the security group to delete
     * @return a CompletableFuture that completes when the security group is deleted
     */
    public CompletableFuture<Void> deleteEC2SecGroupAsync(String groupId) {
        DeleteSecurityGroupRequest request = DeleteSecurityGroupRequest.builder()
            .groupId(groupId)
            .build();
```

```
        CompletableFuture<DeleteSecurityGroupResponse> response =
getAsyncClient().deleteSecurityGroup(request);
        return response.whenComplete((resp, ex) -> {
            if (ex != null) {
                throw new RuntimeException("Failed to delete security group with Id
" + groupId, ex);
            } else if (resp == null) {
                throw new RuntimeException("No response received for deleting
security group with Id " + groupId);
            }
        }).thenApply(resp -> null);
    }
```

- Per i dettagli sull'API, consulta la DeleteSecurityGroupsezione 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.

```
    /**
     * Asynchronously retrieves the instance types available in the current AWS
region.
     * <p>
     * This method uses the AWS SDK's asynchronous API to fetch the available
instance types
     * and then processes the response. It logs the memory information, network
information,
     * and instance type for each instance type returned. Additionally, it returns a
     * {@link CompletableFuture} that resolves to the instance type string for the
"t2.2xlarge"
```

```
    * instance type, if it is found in the response. If the "t2.2xlarge" instance
type is not
    * found, an empty string is returned.
    * </p>
    *
    * @return a {@link CompletableFuture} that resolves to the instance type string
for the
    * "t2.2xlarge" instance type, or an empty string if the instance type is not
found
    */
   public CompletableFuture<String> getInstanceTypesAsync() {
       DescribeInstanceTypesRequest typesRequest =
DescribeInstanceTypesRequest.builder()
           .maxResults(10)
           .build();

       CompletableFuture<DescribeInstanceTypesResponse> response =
getAsyncClient().describeInstanceTypes(typesRequest);
       response.whenComplete((resp, ex) -> {
           if (resp != null) {
               List<InstanceTypeInfo> instanceTypes = resp.instanceTypes();
               for (InstanceTypeInfo type : instanceTypes) {
                   logger.info("The memory information of this type is " +
type.memoryInfo().sizeInMiB());
                   logger.info("Network information is " +
type.networkInfo().toString());
                   logger.info("Instance type is " +
type.instanceType().toString());
               }
           } else {
               throw (RuntimeException) ex;
           }
       });

       return response.thenApply(resp -> {
           for (InstanceTypeInfo type : resp.instanceTypes()) {
               String instanceType = type.instanceType().toString();
               if (instanceType.equals("t2.2xlarge")) {
                   return instanceType;
               }
           }
           return "";
       });
   }
```

- Per i dettagli sull'API, DescribeInstanceTypesconsulta 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.

```
   /**
    * Asynchronously describes an AWS EC2 image with the specified image ID.
    *
    * @param imageId the ID of the image to be described
    * @return a {@link CompletableFuture} that, when completed, contains the ID of
the described image
    * @throws RuntimeException if no images are found with the provided image ID,
or if an error occurs during the AWS API call
    */
   public CompletableFuture<String> describeImageAsync(String imageId) {
       DescribeImagesRequest imagesRequest = DescribeImagesRequest.builder()
           .imageIds(imageId)
           .build();

       AtomicReference<String> imageIdRef = new AtomicReference<>();
       DescribeImagesPublisher paginator =
getAsyncClient().describeImagesPaginator(imagesRequest);
       return paginator.subscribe(response -> {
           response.images().stream()
               .filter(image -> image.imageId().equals(imageId))
               .findFirst()
               .ifPresent(image -> {
                   logger.info("The description of the image is " +
image.description());
                   logger.info("The name of the image is " + image.name());
```

```
                    imageIdRef.set(image.imageId());
                });
        }).thenApply(v -> {
            String id = imageIdRef.get();
            if (id == null) {
                throw new RuntimeException("No images found with the provided image
ID.");
            }
            return id;
        }).exceptionally(ex -> {
            logger.info("Failed to describe image: " + ex.getMessage());
            throw new RuntimeException("Failed to describe image", ex);
        });
    }
```

- Per i dettagli sull'API, DescribeInstancesconsulta 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.

```
    /**
     * Asynchronously describes the key pairs associated with the current AWS
account.
     *
     * @return a {@link CompletableFuture} containing the {@link
DescribeKeyPairsResponse} object, which provides
     * information about the key pairs.
     */
    public CompletableFuture<DescribeKeyPairsResponse> describeKeysAsync() {
        CompletableFuture<DescribeKeyPairsResponse> responseFuture =
getAsyncClient().describeKeyPairs();
```

```
        responseFuture.whenComplete((response, exception) -> {
            if (exception != null) {
              throw new RuntimeException("Failed to describe key pairs: " +
exception.getMessage(), exception);
            }
        });

        return responseFuture;
    }
```

- Per i dettagli sull'API, DescribeKeyPairsconsulta 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.

```
    /**
     * Asynchronously describes the security groups for the specified group ID.
     *
     * @param groupName the name of the security group to describe
     * @return a {@link CompletableFuture} that represents the asynchronous
operation
     *         of describing the security groups. The future will complete with a
     *         {@link DescribeSecurityGroupsResponse} object that contains the
     *         security group information.
     */
    public CompletableFuture<String> describeSecurityGroupArnByNameAsync(String
groupName) {
        DescribeSecurityGroupsRequest request =
DescribeSecurityGroupsRequest.builder()
            .groupNames(groupName)
            .build();
```

```
        DescribeSecurityGroupsPublisher paginator =
getAsyncClient().describeSecurityGroupsPaginator(request);
        AtomicReference<String> groupIdRef = new AtomicReference<>();
        return paginator.subscribe(response -> {
            response.securityGroups().stream()
                .filter(securityGroup ->
securityGroup.groupName().equals(groupName))
                .findFirst()
                .ifPresent(securityGroup ->
groupIdRef.set(securityGroup.groupId()));
        }).thenApply(v -> {
            String groupId = groupIdRef.get();
            if (groupId == null) {
                throw new RuntimeException("No security group found with the name: "
+ groupName);
            }
            return groupId;
        }).exceptionally(ex -> {
            logger.info("Failed to describe security group: " + ex.getMessage());
            throw new RuntimeException("Failed to describe security group", ex);
        });
    }
```

- Per i dettagli sull'API, DescribeSecurityGroupsconsulta 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.

```
    /**
     * Disassociates an Elastic IP address from an instance asynchronously.
     *
```

```
    * @param associationId The ID of the association you want to disassociate.
    * @return a {@link CompletableFuture} representing the asynchronous operation
of disassociating the address. The
    *         {@link CompletableFuture} will complete with a {@link
DisassociateAddressResponse} when the operation is
    *         finished.
    * @throws RuntimeException if the disassociation of the address fails.
    */
   public CompletableFuture<DisassociateAddressResponse>
disassociateAddressAsync(String associationId) {
       Ec2AsyncClient ec2 = getAsyncClient();
       DisassociateAddressRequest addressRequest =
DisassociateAddressRequest.builder()
           .associationId(associationId)
           .build();

       // Disassociate the address asynchronously.
       CompletableFuture<DisassociateAddressResponse> response =
ec2.disassociateAddress(addressRequest);
       response.whenComplete((resp, ex) -> {
           if (ex != null) {
              throw new RuntimeException("Failed to disassociate address", ex);
           }
       });

       return response;
   }
```

- Per i dettagli sull'API, DisassociateAddressconsulta AWS SDK for Java 2.x API Reference.

`GetPasswordData`

Il seguente esempio di codice mostra come utilizzare`GetPasswordData`.

SDK per Java 2.x

> **Note**
>
> C'è 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.Ec2AsyncClient;
import software.amazon.awssdk.services.ec2.model.*;
import java.util.concurrent.CompletableFuture;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class GetPasswordData {

    public static void main(String[] args) {
        final String usage = """

                Usage:
                   <instanceId>

                Where:
                   instanceId - An instance id value that you can obtain from the
 AWS Management Console.\s
             """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }
        String instanceId = args[0];
        Ec2AsyncClient ec2AsyncClient = Ec2AsyncClient.builder()
            .region(Region.US_EAST_1)
            .build();

        try {
            CompletableFuture<Void> future = getPasswordDataAsync(ec2AsyncClient,
 instanceId);
            future.join();
        } catch (RuntimeException rte) {
            System.err.println("An exception occurred: " + (rte.getCause() != null ?
 rte.getCause().getMessage() : rte.getMessage()));
        }
```

```
    }

    /**
     * Fetches the password data for the specified EC2 instance asynchronously.
     *
     * @param ec2AsyncClient the EC2 asynchronous client to use for the request
     * @param instanceId instanceId the ID of the EC2 instance for which you want to
 fetch the password data
     * @return a {@link CompletableFuture} that completes when the password data has
 been fetched
     * @throws RuntimeException if there was a failure in fetching the password data
     */
    public static CompletableFuture<Void> getPasswordDataAsync(Ec2AsyncClient
 ec2AsyncClient, String instanceId) {
        GetPasswordDataRequest getPasswordDataRequest =
 GetPasswordDataRequest.builder()
            .instanceId(instanceId)
            .build();


        CompletableFuture<GetPasswordDataResponse> response =
 ec2AsyncClient.getPasswordData(getPasswordDataRequest);
        response.whenComplete((getPasswordDataResponse, ex) -> {
            if (ex != null) {
                throw new RuntimeException("Failed to get password data for
 instance: " + instanceId, ex);
            } else if (getPasswordDataResponse == null ||
 getPasswordDataResponse.passwordData().isEmpty()) {
                throw new RuntimeException("No password data found for instance: " +
 instanceId);
            } else {
                String encryptedPasswordData =
 getPasswordDataResponse.passwordData();
                System.out.println("Encrypted Password Data: " +
 encryptedPasswordData);
            }
        });

        return response.thenApply(resp -> null);
    }
}
```

- Per i dettagli sull'API, GetPasswordDataconsulta 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.

```
   /**
    * Releases an Elastic IP address asynchronously.
    *
    * @param allocId the allocation ID of the Elastic IP address to be released
    * @return a {@link CompletableFuture} representing the asynchronous operation
of releasing the Elastic IP address
    */
   public CompletableFuture<ReleaseAddressResponse> releaseEC2AddressAsync(String
allocId) {
       ReleaseAddressRequest request = ReleaseAddressRequest.builder()
           .allocationId(allocId)
           .build();

       CompletableFuture<ReleaseAddressResponse> response =
getAsyncClient().releaseAddress(request);
       response.whenComplete((resp, ex) -> {
           if (ex != null) {
               throw new RuntimeException("Failed to release Elastic IP address",
ex);
           }
       });

       return response;
   }
```

- Per i dettagli sull'API, ReleaseAddressconsulta 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.

```
   /**
    * Runs an EC2 instance asynchronously.
    *
    * @param instanceType The instance type to use for the EC2 instance.
    * @param keyName The name of the key pair to associate with the EC2 instance.
    * @param groupName The name of the security group to associate with the EC2
instance.
    * @param amiId The ID of the Amazon Machine Image (AMI) to use for the EC2
instance.
    * @return A {@link CompletableFuture} that completes with the ID of the started
EC2 instance.
    * @throws RuntimeException If there is an error running the EC2 instance.
    */
   public CompletableFuture<String> runInstanceAsync(String instanceType, String
keyName, String groupName, String amiId) {
       RunInstancesRequest runRequest = RunInstancesRequest.builder()
           .instanceType(instanceType)
           .keyName(keyName)
           .securityGroups(groupName)
           .maxCount(1)
           .minCount(1)
           .imageId(amiId)
           .build();

       CompletableFuture<RunInstancesResponse> responseFuture =
getAsyncClient().runInstances(runRequest);
       return responseFuture.thenCompose(response -> {
           String instanceIdVal = response.instances().get(0).instanceId();
           System.out.println("Going to start an EC2 instance and use a waiter to
wait for it to be in running state");
           return getAsyncClient().waiter()
```

```
                .waitUntilInstanceExists(r -> r.instanceIds(instanceIdVal))
                .thenCompose(waitResponse -> getAsyncClient().waiter()
                    .waitUntilInstanceRunning(r -> r.instanceIds(instanceIdVal))
                    .thenApply(runningResponse -> instanceIdVal));
        }).exceptionally(throwable -> {
            // Handle any exceptions that occurred during the async call
            throw new RuntimeException("Failed to run EC2 instance: " +
throwable.getMessage(), throwable);
        });
    }
```

- Per i dettagli sull'API, RunInstancesconsulta 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.

```
    /**
     * Starts an Amazon EC2 instance asynchronously and waits until it is in the
"running" state.
     *
     * @param instanceId the ID of the instance to start
     * @return a {@link CompletableFuture} that completes when the instance has been
started and is in the "running" state, or exceptionally if an error occurs
     */
    public CompletableFuture<Void> startInstanceAsync(String instanceId) {
        StartInstancesRequest startRequest = StartInstancesRequest.builder()
            .instanceIds(instanceId)
            .build();

        Ec2AsyncWaiter ec2Waiter = Ec2AsyncWaiter.builder()
            .client(getAsyncClient())
            .build();
```

```
        DescribeInstancesRequest describeRequest =
DescribeInstancesRequest.builder()
            .instanceIds(instanceId)
            .build();

        logger.info("Starting instance " + instanceId + " and waiting for it to
run.");
        CompletableFuture<Void> resultFuture = new CompletableFuture<>();
        return getAsyncClient().startInstances(startRequest)
            .thenCompose(response ->
                ec2Waiter.waitUntilInstanceRunning(describeRequest)
            )
            .thenAccept(waiterResponse -> {
                logger.info("Successfully started instance " + instanceId);
                resultFuture.complete(null);
            })
            .exceptionally(throwable -> {
                resultFuture.completeExceptionally(new RuntimeException("Failed to
start instance: " + throwable.getMessage(), throwable));
                return null;
            });
    }
```

- Per i dettagli sull'API, StartInstancesconsulta 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.

```
    /**
     * Stops the EC2 instance with the specified ID asynchronously and waits for the
instance to stop.
```

```
     *
     * @param instanceId the ID of the EC2 instance to stop
     * @return a {@link CompletableFuture} that completes when the instance has been
stopped, or exceptionally if an error occurs
     */
    public CompletableFuture<Void> stopInstanceAsync(String instanceId) {
        StopInstancesRequest stopRequest = StopInstancesRequest.builder()
            .instanceIds(instanceId)
            .build();

        DescribeInstancesRequest describeRequest =
DescribeInstancesRequest.builder()
            .instanceIds(instanceId)
            .build();

        Ec2AsyncWaiter ec2Waiter = Ec2AsyncWaiter.builder()
            .client(getAsyncClient())
            .build();

        CompletableFuture<Void> resultFuture = new CompletableFuture<>();
        logger.info("Stopping instance " + instanceId + " and waiting for it to
stop.");
        getAsyncClient().stopInstances(stopRequest)
            .thenCompose(response -> {
                if (response.stoppingInstances().isEmpty()) {
                    return CompletableFuture.failedFuture(new RuntimeException("No
instances were stopped. Please check the instance ID: " + instanceId));
                }
                return ec2Waiter.waitUntilInstanceStopped(describeRequest);
            })
            .thenAccept(waiterResponse -> {
                logger.info("Successfully stopped instance " + instanceId);
                resultFuture.complete(null);
            })
            .exceptionally(throwable -> {
                logger.error("Failed to stop instance " + instanceId + ": " +
throwable.getMessage(), throwable);
                resultFuture.completeExceptionally(new RuntimeException("Failed to
stop instance: " + throwable.getMessage(), throwable));
                return null;
            });

        return resultFuture;
    }
```

- Per i dettagli sull'API, StopInstancesconsulta 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.

```
   /**
    * Terminates an EC2 instance asynchronously and waits for it to reach the
terminated state.
    *
    * @param instanceId the ID of the EC2 instance to terminate
    * @return a {@link CompletableFuture} that completes when the instance has been
terminated
    * @throws RuntimeException if there is no response from the AWS SDK or if there
is a failure during the termination process
    */
   public CompletableFuture<Object> terminateEC2Async(String instanceId) {
       TerminateInstancesRequest terminateRequest =
TerminateInstancesRequest.builder()
           .instanceIds(instanceId)
           .build();

       CompletableFuture<TerminateInstancesResponse> responseFuture =
getAsyncClient().terminateInstances(terminateRequest);
       return responseFuture.thenCompose(terminateResponse -> {
           if (terminateResponse == null) {
               throw new RuntimeException("No response received for terminating
instance " + instanceId);
           }
           System.out.println("Going to terminate an EC2 instance and use a waiter
to wait for it to be in terminated state");
```

```
            return getAsyncClient().waiter()
                .waitUntilInstanceTerminated(r -> r.instanceIds(instanceId))
                .thenApply(waiterResponse -> null);
        }).exceptionally(throwable -> {
            // Handle any exceptions that occurred during the async call
            throw new RuntimeException("Failed to terminate EC2 instance: " +
throwable.getMessage(), throwable);
        });
    }
```

- Per i dettagli sull'API, TerminateInstancesconsulta 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 Amazon EC2 Auto Scaling per creare istanze Amazon Elastic Compute Cloud (Amazon EC2) basate su un modello di avvio e per mantenere il numero di istanze in un intervallo specificato.
- Gestisci e distribuisci 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 EC2 istanza 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 di integrità aggiornando AWS Systems Manager i parametri.

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.

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 wasSuccessul = loadBalancer.verifyLoadBalancerEndpoint(elbDnsName);
        if (!wasSuccessul) {
            System.out.println("Couldn't connect to the load balancer, verifying
that the port is open...");
            CloseableHttpClient httpClient = HttpClients.createDefault();

            // Create an HTTP GET request to "http://checkip.amazonaws.com"
            HttpGet httpGet = new HttpGet("http://checkip.amazonaws.com");
            try {
                // Execute the request and get the response
                HttpResponse response = httpClient.execute(httpGet);

                // Read the response content.
                String ipAddress =
IOUtils.toString(response.getEntity().getContent(), StandardCharsets.UTF_8).trim();

                // Print the public IP address.
                System.out.println("Public IP Address: " + ipAddress);
                GroupInfo groupInfo = autoScaler.verifyInboundPort(vpcId, port,
ipAddress);
                if (!groupInfo.isPortOpen()) {
                    System.out.println("""
                            For this example to work, the default security group for
your default VPC must
                            allow access from this computer. You can either add it
automatically from this
                            example or add it yourself using the AWS Management
Console.
                            """);

                    System.out.println(
                            "Do you want to add a rule to security group " +
groupInfo.getGroupName() + " to allow");
                    System.out.println("inbound traffic on port " + port + " from
your computer's IP address (y/n) ");
                    String ans = in.nextLine();
                    if ("y".equalsIgnoreCase(ans)) {
                        autoScaler.openInboundPort(groupInfo.getGroupName(),
String.valueOf(port), ipAddress);
                        System.out.println("Security group rule added.");
                    } else {
```

```
                        System.out.println("No security group rule added.");
                    }
                }

            } catch (AutoScalingException e) {
                e.printStackTrace();
            }
        } else if (wasSuccessul) {
            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 azioni di Auto Scaling e EC2 Amazon.

```
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.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)
                    .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

Esempi di Amazon ECR con SDK for Java 2.x

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando Amazon ECR. AWS SDK for Java 2.x

Le nozioni di base sono esempi di codice che mostrano come eseguire le operazioni essenziali all'interno di un servizio.

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.

Ogni esempio include un collegamento al codice sorgente completo, dove puoi trovare istruzioni su come configurare ed eseguire il codice nel contesto.

Nozioni di base

Salve Amazon ECR

I seguenti esempi di codice mostrano come iniziare a usare Amazon ECR.

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.ecr.EcrClient;
import software.amazon.awssdk.services.ecr.model.EcrException;
import software.amazon.awssdk.services.ecr.model.ListImagesRequest;
import software.amazon.awssdk.services.ecr.paginators.ListImagesIterable;

public class HelloECR {

    public static void main(String[] args) {
        final String usage = """
            Usage:    <repositoryName>

            Where:
```

```
                repositoryName - The name of the Amazon ECR repository.
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String repoName = args[0];
        EcrClient ecrClient = EcrClient.builder()
            .region(Region.US_EAST_1)
            .build();

        listImageTags(ecrClient, repoName);
    }
    public static void listImageTags(EcrClient ecrClient, String repoName){
        ListImagesRequest listImagesPaginator = ListImagesRequest.builder()
            .repositoryName(repoName)
            .build();

        ListImagesIterable imagesIterable =
 ecrClient.listImagesPaginator(listImagesPaginator);
        imagesIterable.stream()
            .flatMap(r -> r.imageIds().stream())
            .forEach(image -> System.out.println("The docker image tag is: "
 +image.imageTag()));
    }
}
```

- Per i dettagli sull'API, consulta ListImages in AWS SDK for Java 2.x API Reference.

Argomenti

- Nozioni di base
- Azioni

Nozioni di base

Informazioni di base

L'esempio di codice seguente mostra come:

- Crea un repository Amazon ECR.
- Imposta le politiche del repository.
- Recupera il repository. URIs
- Ottieni i token di autorizzazione Amazon ECR.
- Imposta le politiche del ciclo di vita per i repository Amazon ECR.
- Invia un'immagine Docker a un repository Amazon ECR.
- Verifica l'esistenza di un'immagine in un repository Amazon ECR.
- Elenca i repository Amazon ECR per il tuo account e ottieni dettagli su di essi.
- Elimina i repository Amazon ECR.

SDK per Java 2.x

> **Note**
>
> C'è altro da fare. 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 caratteristiche di Amazon ECR.

```
import software.amazon.awssdk.services.ecr.model.EcrException;
import software.amazon.awssdk.services.ecr.model.RepositoryPolicyNotFoundException;

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
 *
 * This Java code example requires an IAM Role that has permissions to interact with
 the Amazon ECR service.
 *
 * To create an IAM role, see:
 *
```

```
 * https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles_create.html
 *
 * This Java scenario example requires a local docker image named echo-text. Without
 a local image,
 * this Java program will not successfully run. For more information including how
 to create the local
 * image, see:
 *
 * /scenarios/basics/ecr/README
 *
 */
public class ECRScenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");
    public static void main(String[] args) {
        final String usage = """
            Usage: <iamRoleARN> <accountId>

            Where:
               iamRoleARN - The IAM role ARN that has the necessary permissions to
 access and manage the Amazon ECR repository.
               accountId - Your AWS account number.
            """;

        if (args.length != 2) {
            System.out.println(usage);
            return;
        }

        ECRActions ecrActions = new ECRActions();
        String iamRole = args[0];
        String accountId = args[1];
        String localImageName;

        Scanner scanner = new Scanner(System.in);
        System.out.println("""
             The Amazon Elastic Container Registry (ECR) is a fully-managed Docker
 container registry
             service provided by AWS. It allows developers and organizations to
 securely
             store, manage, and deploy Docker container images.
             ECR provides a simple and scalable way to manage container images
 throughout their lifecycle,
             from building and testing to production deployment.\s
```

```
            The `EcrAsyncClient` interface in the AWS SDK for Java 2.x provides a
set of methods to
            programmatically interact with the Amazon ECR service. This allows
developers to
            automate the storage, retrieval, and management of container images as
part of their application
            deployment pipelines. With ECR, teams can focus on building and
deploying their
            applications without having to worry about the underlying
infrastructure required to
            host and manage a container registry.

           This scenario walks you through how to perform key operations for this
service.
           Let's get started...

           You have two choices:
           1 - Run the entire program.
           2 - Delete an existing Amazon ECR repository named echo-text (created
from a previous execution of
           this program that did not complete).
           """);

       while (true) {
           String input = scanner.nextLine();
           if (input.trim().equalsIgnoreCase("1")) {
               System.out.println("Continuing with the program...");
               System.out.println("");
               break;
           } else if (input.trim().equalsIgnoreCase("2")) {
               String repoName = "echo-text";
               ecrActions.deleteECRRepository(repoName);
               return;
           } else {
               // Handle invalid input.
               System.out.println("Invalid input. Please try again.");
           }
       }

       waitForInputToContinue(scanner);
       System.out.println(DASHES);

       System.out.println("""
          1. Create an ECR repository.
```

```
           The first task is to ensure we have a local Docker image named echo-
text.
           If this image exists, then an Amazon ECR repository is created.

           An ECR repository is a private Docker container repository provided
           by Amazon Web Services (AWS). It is a managed service that makes it easy
           to store, manage, and deploy Docker container images.\s
           """ );

        // Ensure that a local docker image named echo-text exists.
        boolean doesExist = ecrActions.isEchoTextImagePresent();
        String repoName;
        if (!doesExist){
            System.out.println("The local image named echo-text does not exist");
            return;
        } else {
            localImageName = "echo-text";
            repoName = "echo-text";
        }

        try {
            String repoArn = ecrActions.createECRRepository(repoName);
            System.out.println("The ARN of the ECR repository is " + repoArn);

        } catch (IllegalArgumentException e) {
            System.err.println("Invalid repository name: " + e.getMessage());
            return;
        } catch (RuntimeException e) {
            System.err.println("An error occurred while creating the ECR repository:
 " + e.getMessage());
            e.printStackTrace();
            return;
        }
        waitForInputToContinue(scanner);

        System.out.println(DASHES);
        System.out.println("""
        2. Set an ECR repository policy.

        Setting an ECR repository policy using the `setRepositoryPolicy` function is
 crucial for maintaining
        the security and integrity of your container images. The repository policy
 allows you to
```

```
        define specific rules and restrictions for accessing and managing the images
stored within your ECR
        repository.
        """);
        waitForInputToContinue(scanner);
        try {
            ecrActions.setRepoPolicy(repoName, iamRole);

        } catch (RepositoryPolicyNotFoundException e) {
            System.err.println("Invalid repository name: " + e.getMessage());
            return;
        } catch (EcrException e) {
            System.err.println("An ECR exception occurred: " + e.getMessage());
            return;
        } catch (RuntimeException e) {
            System.err.println("An error occurred while creating the ECR repository:
" + e.getMessage());
            return;
        }
        waitForInputToContinue(scanner);

        System.out.println(DASHES);
        System.out.println("""
        3. Display ECR repository policy.

        Now we will retrieve the ECR policy to ensure it was successfully set.
        """);
        waitForInputToContinue(scanner);
        try {
            String policyText = ecrActions.getRepoPolicy(repoName);
            System.out.println("Policy Text:");
            System.out.println(policyText);

        } catch (EcrException e) {
            System.err.println("An ECR exception occurred: " + e.getMessage());
            return;
        } catch (RuntimeException e) {
            System.err.println("An error occurred while creating the ECR repository:
" + e.getMessage());
            return;
        }

        waitForInputToContinue(scanner);
```

```
        System.out.println(DASHES);
        System.out.println("""
        4. Retrieve an ECR authorization token.

        You need an authorization token to securely access and interact with the
Amazon ECR registry.
        The `getAuthorizationToken` method of the `EcrAsyncClient` is responsible
for securely accessing
        and interacting with an Amazon ECR repository. This operation is responsible
for obtaining a
        valid authorization token, which is required to authenticate your requests
to the ECR service.

        Without a valid authorization token, you would not be able to perform any
operations on the
        ECR repository, such as pushing, pulling, or managing your Docker images.

        """);
        waitForInputToContinue(scanner);
        try {
             ecrActions.getAuthToken();

        } catch (EcrException e) {
            System.err.println("An ECR exception occurred: " + e.getMessage());
            return;
        } catch (RuntimeException e) {
            System.err.println("An error occurred while retrieving the authorization
token: " + e.getMessage());
            return;
        }
        waitForInputToContinue(scanner);

        System.out.println(DASHES);
        System.out.println("""
        5. Get the ECR Repository URI.

        The URI  of an Amazon ECR repository is important. When you want to deploy a
container image to
        a container orchestration platform like Amazon Elastic Kubernetes Service
(EKS)
        or Amazon Elastic Container Service (ECS), you need to specify the full
image URI,
        which includes the ECR repository URI. This allows the container runtime to
pull the
```

```
        correct container image from the ECR repository.
       """);
        waitForInputToContinue(scanner);

        try {
            ecrActions.getRepositoryURI(repoName);

        } catch (EcrException e) {
            System.err.println("An ECR exception occurred: " + e.getMessage());
            return;

        } catch (RuntimeException e) {
            System.err.println("An error occurred while retrieving the URI: " +
e.getMessage());
            return;
        }
        waitForInputToContinue(scanner);

        System.out.println(DASHES);
        System.out.println("""
            6. Set an ECR Lifecycle Policy.

            An ECR Lifecycle Policy is used to manage the lifecycle of Docker images
stored in your ECR repositories.
            These policies allow you to automatically remove old or unused Docker
images from your repositories,
            freeing up storage space and reducing costs.

            This example policy helps to maintain the size and efficiency of the
container registry
            by automatically removing older and potentially unused images, ensuring
that the
            storage is optimized and the registry remains up-to-date.
            """);
        waitForInputToContinue(scanner);
        try {
            ecrActions.setLifeCyclePolicy(repoName);

        } catch (RuntimeException e) {
            System.err.println("An error occurred while setting the lifecycle
policy: " + e.getMessage());
            e.printStackTrace();
            return;
        }
```

```
        waitForInputToContinue(scanner);

        System.out.println(DASHES);
        System.out.println("""
        7. Push a docker image to the Amazon ECR Repository.

        The `pushImageCmd()` method pushes a local Docker image to an Amazon ECR
repository.
        It sets up the Docker client by connecting to the local Docker host using
the default port.
        It then retrieves the authorization token for the ECR repository by making a
call to the AWS SDK.

        The method uses the authorization token to create an `AuthConfig` object,
which is used to authenticate
        the Docker client when pushing the image. Finally, the method tags the
Docker image with the specified
        repository name and image tag, and then pushes the image to the ECR
repository using the Docker client.
        If the push operation is successful, the method prints a message indicating
that the image was pushed to ECR.
        """);
        waitForInputToContinue(scanner);

        try {
            ecrActions.pushDockerImage(repoName, localImageName);

        } catch (RuntimeException e) {
            System.err.println("An error occurred while pushing a local Docker image
to Amazon ECR: " + e.getMessage());
            e.printStackTrace();
            return;
        }
        waitForInputToContinue(scanner);

        System.out.println(DASHES);
        System.out.println("8. Verify if the image is in the ECR Repository.");
        waitForInputToContinue(scanner);
        try {
            ecrActions.verifyImage(repoName, localImageName);

        } catch (EcrException e) {
            System.err.println("An ECR exception occurred: " + e.getMessage());
            return;
```

```
        } catch (RuntimeException e) {
            System.err.println("An error occurred " + e.getMessage());
            e.printStackTrace();
            return;
        }
        waitForInputToContinue(scanner);

        System.out.println(DASHES);
        System.out.println("9. As an optional step, you can interact with the image
 in Amazon ECR by using the CLI.");
        System.out.println("Would you like to view instructions on how to use the
 CLI to run the image? (y/n)");
        String ans = scanner.nextLine().trim();
        if (ans.equalsIgnoreCase("y")) {
            String instructions = """
            1. Authenticate with ECR - Before you can pull the image from Amazon
 ECR, you need to authenticate with the registry. You can do this using the AWS CLI:

                aws ecr get-login-password --region us-east-1 | docker login --
username AWS --password-stdin %s.dkr.ecr.us-east-1.amazonaws.com

            2. Describe the image using this command:

               aws ecr describe-images --repository-name %s --image-ids imageTag=%s

            3. Run the Docker container and view the output using this command:

               docker run --rm %s.dkr.ecr.us-east-1.amazonaws.com/%s:%s
            """;

            instructions = String.format(instructions, accountId, repoName,
 localImageName, accountId, repoName, localImageName);
            System.out.println(instructions);
        }
        waitForInputToContinue(scanner);

        System.out.println(DASHES);
        System.out.println("10. Delete the ECR Repository.");
        System.out.println(
        """
        If the repository isn't empty, you must either delete the contents of the
 repository
        or use the force option (used in this scenario) to delete the repository and
 have Amazon ECR delete all of its contents
```

```
        on your behalf.
        """);
        System.out.println("Would you like to delete the Amazon ECR Repository? (y/
n)");
        String delAns = scanner.nextLine().trim();
        if (delAns.equalsIgnoreCase("y")) {
            System.out.println("You selected to delete the AWS ECR resources.");

            try {
                ecrActions.deleteECRRepository(repoName);

            } catch (EcrException e) {
                System.err.println("An ECR exception occurred: " + e.getMessage());
                return;
            } catch (RuntimeException e) {
                System.err.println("An error occurred while deleting the Docker
 image: " + e.getMessage());
                e.printStackTrace();
                return;
            }
        }

        System.out.println(DASHES);
        System.out.println("This concludes the Amazon ECR SDK scenario");
        System.out.println(DASHES);
    }

   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.");
           }
       }
   }
}
```

Una classe wrapper per i metodi Amazon ECR SDK.

```
import com.github.dockerjava.api.DockerClient;
import com.github.dockerjava.api.exception.DockerClientException;
import com.github.dockerjava.api.model.AuthConfig;
import com.github.dockerjava.api.model.Image;
import com.github.dockerjava.core.DockerClientBuilder;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.client.config.ClientOverrideConfiguration;
import software.amazon.awssdk.http.async.SdkAsyncHttpClient;
import software.amazon.awssdk.http.nio.netty.NettyNioAsyncHttpClient;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ecr.EcrAsyncClient;
import software.amazon.awssdk.services.ecr.model.AuthorizationData;
import software.amazon.awssdk.services.ecr.model.CreateRepositoryRequest;
import software.amazon.awssdk.services.ecr.model.CreateRepositoryResponse;
import software.amazon.awssdk.services.ecr.model.DeleteRepositoryRequest;
import software.amazon.awssdk.services.ecr.model.DeleteRepositoryResponse;
import software.amazon.awssdk.services.ecr.model.DescribeImagesRequest;
import software.amazon.awssdk.services.ecr.model.DescribeImagesResponse;
import software.amazon.awssdk.services.ecr.model.DescribeRepositoriesRequest;
import software.amazon.awssdk.services.ecr.model.DescribeRepositoriesResponse;
import software.amazon.awssdk.services.ecr.model.EcrException;
import software.amazon.awssdk.services.ecr.model.GetAuthorizationTokenResponse;
import software.amazon.awssdk.services.ecr.model.GetRepositoryPolicyRequest;
import software.amazon.awssdk.services.ecr.model.GetRepositoryPolicyResponse;
import software.amazon.awssdk.services.ecr.model.ImageIdentifier;
import software.amazon.awssdk.services.ecr.model.Repository;
import software.amazon.awssdk.services.ecr.model.RepositoryPolicyNotFoundException;
import software.amazon.awssdk.services.ecr.model.SetRepositoryPolicyRequest;
import software.amazon.awssdk.services.ecr.model.SetRepositoryPolicyResponse;
import software.amazon.awssdk.services.ecr.model.StartLifecyclePolicyPreviewRequest;
import
 software.amazon.awssdk.services.ecr.model.StartLifecyclePolicyPreviewResponse;
import com.github.dockerjava.api.command.DockerCmdExecFactory;
import com.github.dockerjava.netty.NettyDockerCmdExecFactory;
import java.time.Duration;
import java.util.Base64;
import java.util.List;
import java.util.concurrent.CompletableFuture;
```

```
import java.util.concurrent.CompletionException;

public class ECRActions {
    private static EcrAsyncClient ecrClient;

    private static DockerClient dockerClient;

    private static Logger logger = LoggerFactory.getLogger(ECRActions.class);

    /**
     * Creates an Amazon Elastic Container Registry (Amazon ECR) repository.
     *
     * @param repoName the name of the repository to create.
     * @return the Amazon Resource Name (ARN) of the created repository, or an empty
 string if the operation failed.
     * @throws IllegalArgumentException     If repository name is invalid.
     * @throws RuntimeException             if an error occurs while creating the
 repository.
     */
    public String createECRRepository(String repoName) {
        if (repoName == null || repoName.isEmpty()) {
            throw new IllegalArgumentException("Repository name cannot be null or
 empty");
        }

        CreateRepositoryRequest request = CreateRepositoryRequest.builder()
            .repositoryName(repoName)
            .build();

        CompletableFuture<CreateRepositoryResponse> response =
 getAsyncClient().createRepository(request);
        try {
            CreateRepositoryResponse result = response.join();
            if (result != null) {
                System.out.println("The " + repoName + " repository was created
 successfully.");
                return result.repository().repositoryArn();
            } else {
                throw new RuntimeException("Unexpected response type");
            }
        } catch (CompletionException e) {
            Throwable cause = e.getCause();
            if (cause instanceof EcrException ex) {
```

```
                if
("RepositoryAlreadyExistsException".equals(ex.awsErrorDetails().errorCode())) {
                    System.out.println("The Amazon ECR repository already exists,
moving on...");
                    DescribeRepositoriesRequest describeRequest =
DescribeRepositoriesRequest.builder()
                        .repositoryNames(repoName)
                        .build();
                    DescribeRepositoriesResponse describeResponse =
getAsyncClient().describeRepositories(describeRequest).join();
                    return describeResponse.repositories().get(0).repositoryArn();
                } else {
                    throw new RuntimeException(ex);
                }
            } else {
                throw new RuntimeException(e);
            }
        }
    }

    /**
     * Deletes an ECR (Elastic Container Registry) repository.
     *
     * @param repoName the name of the repository to delete.
     * @throws IllegalArgumentException if the repository name is null or empty.
     * @throws EcrException if there is an error deleting the repository.
     * @throws RuntimeException if an unexpected error occurs during the deletion
process.
     */
    public void deleteECRRepository(String repoName) {
        if (repoName == null || repoName.isEmpty()) {
            throw new IllegalArgumentException("Repository name cannot be null or
empty");
        }

        DeleteRepositoryRequest repositoryRequest =
DeleteRepositoryRequest.builder()
            .force(true)
            .repositoryName(repoName)
            .build();

        CompletableFuture<DeleteRepositoryResponse> response =
getAsyncClient().deleteRepository(repositoryRequest);
        response.whenComplete((deleteRepositoryResponse, ex) -> {
```

```
            if (deleteRepositoryResponse != null) {
                System.out.println("You have successfully deleted the " + repoName +
" repository");
            } else {
                Throwable cause = ex.getCause();
                if (cause instanceof EcrException) {
                    throw (EcrException) cause;
                } else {
                    throw new RuntimeException("Unexpected error: " +
cause.getMessage(), cause);
                }
            }
        });

        // Wait for the CompletableFuture to complete
        response.join();
    }



    private static DockerClient getDockerClient() {
        String osName = System.getProperty("os.name");
        if (osName.startsWith("Windows")) {
            // Make sure Docker Desktop is running.
            String dockerHost = "tcp://localhost:2375"; // Use the Docker Desktop
default port.
            DockerCmdExecFactory dockerCmdExecFactory = new
NettyDockerCmdExecFactory().withReadTimeout(20000).withConnectTimeout(20000);
            dockerClient =
DockerClientBuilder.getInstance(dockerHost).withDockerCmdExecFactory(dockerCmdExecFactory).
        } else {
            dockerClient = DockerClientBuilder.getInstance().build();
        }
        return dockerClient;
    }

    /**
     * Retrieves an asynchronous Amazon Elastic Container Registry (ECR) client.
     *
     * @return the configured ECR asynchronous client.
     */
    private static EcrAsyncClient getAsyncClient() {

        /*
```

```
        The `NettyNioAsyncHttpClient` class is part of the AWS SDK for Java,
version 2,
        and it is designed to provide a high-performance, asynchronous HTTP client
for interacting with AWS services.
        It uses the Netty framework to handle the underlying network communication
and the Java NIO API to
        provide a non-blocking, event-driven approach to HTTP requests and
responses.
        */
       SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
           .maxConcurrency(50)  // Adjust as needed.
           .connectionTimeout(Duration.ofSeconds(60))  // Set the connection
timeout.
           .readTimeout(Duration.ofSeconds(60))  // Set the read timeout.
           .writeTimeout(Duration.ofSeconds(60))  // Set the write timeout.
           .build();

       ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
           .apiCallTimeout(Duration.ofMinutes(2))  // Set the overall API call
timeout.
           .apiCallAttemptTimeout(Duration.ofSeconds(90))  // Set the individual
call attempt timeout.
           .build();

       if (ecrClient == null) {
           ecrClient = EcrAsyncClient.builder()
               .region(Region.US_EAST_1)
               .httpClient(httpClient)
               .overrideConfiguration(overrideConfig)
               .build();
       }
       return ecrClient;
   }

   /**
    * Sets the lifecycle policy for the specified repository.
    *
    * @param repoName the name of the repository for which to set the lifecycle
policy.
    */
   public void setLifeCyclePolicy(String repoName) {
       /*
```

```
           This policy helps to maintain the size and efficiency of the container
 registry
           by automatically removing older and potentially unused images,
           ensuring that the storage is optimized and the registry remains up-to-
date.
         */
        String polText = """
             {
             "rules": [
                 {
                     "rulePriority": 1,
                     "description": "Expire images older than 14 days",
                     "selection": {
                         "tagStatus": "any",
                         "countType": "sinceImagePushed",
                         "countUnit": "days",
                         "countNumber": 14
                     },
                     "action": {
                         "type": "expire"
                     }
                 }
             ]
             }
             """;

        StartLifecyclePolicyPreviewRequest lifecyclePolicyPreviewRequest =
 StartLifecyclePolicyPreviewRequest.builder()
             .lifecyclePolicyText(polText)
             .repositoryName(repoName)
             .build();

        CompletableFuture<StartLifecyclePolicyPreviewResponse> response =
 getAsyncClient().startLifecyclePolicyPreview(lifecyclePolicyPreviewRequest);
        response.whenComplete((lifecyclePolicyPreviewResponse, ex) -> {
            if (lifecyclePolicyPreviewResponse != null) {
                System.out.println("Lifecycle policy preview started
 successfully.");
            } else {
                if (ex.getCause() instanceof EcrException) {
                    throw (EcrException) ex.getCause();
                } else {
                    String errorMessage = "Unexpected error occurred: " +
 ex.getMessage();
```

```
                    throw new RuntimeException(errorMessage, ex);
                }
            }
        });
        // Wait for the CompletableFuture to complete.
        response.join();
    }

    /**
     * Verifies the existence of an image in an Amazon Elastic Container Registry
 (Amazon ECR) repository asynchronously.
     *
     * @param repositoryName The name of the Amazon ECR repository.
     * @param imageTag       The tag of the image to verify.
     * @throws EcrException             if there is an error retrieving the image
 information from Amazon ECR.
     * @throws CompletionException      if the asynchronous operation completes
 exceptionally.
     */
    public void verifyImage(String repositoryName, String imageTag) {
        DescribeImagesRequest request = DescribeImagesRequest.builder()
            .repositoryName(repositoryName)
            .imageIds(ImageIdentifier.builder().imageTag(imageTag).build())
            .build();

        CompletableFuture<DescribeImagesResponse> response =
 getAsyncClient().describeImages(request);
        response.whenComplete((describeImagesResponse, ex) -> {
            if (ex != null) {
                if (ex instanceof CompletionException) {
                    Throwable cause = ex.getCause();
                    if (cause instanceof EcrException) {
                        throw (EcrException) cause;
                    } else {
                        throw new RuntimeException("Unexpected error: " +
 cause.getMessage(), cause);
                    }
                } else {
                    throw new RuntimeException("Unexpected error: " +
 ex.getCause());
                }
            } else if (describeImagesResponse != null &&
 !describeImagesResponse.imageDetails().isEmpty()) {
                System.out.println("Image is present in the repository.");
```

```
            } else {
                System.out.println("Image is not present in the repository.");
            }
        });

        // Wait for the CompletableFuture to complete.
        response.join();
    }

    /**
     * Retrieves the repository URI for the specified repository name.
     *
     * @param repoName the name of the repository to retrieve the URI for.
     * @return the repository URI for the specified repository name.
     * @throws EcrException        if there is an error retrieving the repository
information.
     * @throws CompletionException if the asynchronous operation completes
exceptionally.
     */
    public void getRepositoryURI(String repoName) {
        DescribeRepositoriesRequest request = DescribeRepositoriesRequest.builder()
            .repositoryNames(repoName)
            .build();

        CompletableFuture<DescribeRepositoriesResponse> response =
getAsyncClient().describeRepositories(request);
        response.whenComplete((describeRepositoriesResponse, ex) -> {
            if (ex != null) {
                Throwable cause = ex.getCause();
                if (cause instanceof InterruptedException) {
                    Thread.currentThread().interrupt();
                    String errorMessage = "Thread interrupted while waiting for
asynchronous operation: " + cause.getMessage();
                    throw new RuntimeException(errorMessage, cause);
                } else if (cause instanceof EcrException) {
                    throw (EcrException) cause;
                } else {
                    String errorMessage = "Unexpected error: " + cause.getMessage();
                    throw new RuntimeException(errorMessage, cause);
                }
            } else {
                if (describeRepositoriesResponse != null) {
                    if (!describeRepositoriesResponse.repositories().isEmpty()) {
```

```
                        String repositoryUri =
describeRepositoriesResponse.repositories().get(0).repositoryUri();
                        System.out.println("Repository URI found: " +
repositoryUri);
                    } else {
                        System.out.println("No repositories found for the given
name.");
                    }
                } else {
                    System.err.println("No response received from
describeRepositories.");
                }
            }
        });
        response.join();
    }

    /**
     * Retrieves the authorization token for Amazon Elastic Container Registry
(ECR).
     * This method makes an asynchronous call to the ECR client to retrieve the
authorization token.
     * If the operation is successful, the method prints the token to the console.
     * If an exception occurs, the method handles the exception and prints the error
message.
     *
     * @throws EcrException     if there is an error retrieving the authorization
token from ECR.
     * @throws RuntimeException if there is an unexpected error during the
operation.
     */
    public void getAuthToken() {
        CompletableFuture<GetAuthorizationTokenResponse> response =
getAsyncClient().getAuthorizationToken();
        response.whenComplete((authorizationTokenResponse, ex) -> {
            if (authorizationTokenResponse != null) {
                AuthorizationData authorizationData =
authorizationTokenResponse.authorizationData().get(0);
                String token = authorizationData.authorizationToken();
                if (!token.isEmpty()) {
                    System.out.println("The token was successfully retrieved.");
                }
            } else {
                if (ex.getCause() instanceof EcrException) {
```

```
                    throw (EcrException) ex.getCause();
                } else {
                    String errorMessage = "Unexpected error occurred: " +
ex.getMessage();
                    throw new RuntimeException(errorMessage, ex); // Rethrow the
exception
                }
            }
        });
        response.join();
    }

    /**
     * Gets the repository policy for the specified repository.
     *
     * @param repoName the name of the repository.
     * @throws EcrException if an AWS error occurs while getting the repository
policy.
     */
    public String getRepoPolicy(String repoName) {
        if (repoName == null || repoName.isEmpty()) {
            throw new IllegalArgumentException("Repository name cannot be null or
empty");
        }

        GetRepositoryPolicyRequest getRepositoryPolicyRequest =
GetRepositoryPolicyRequest.builder()
            .repositoryName(repoName)
            .build();

        CompletableFuture<GetRepositoryPolicyResponse> response =
getAsyncClient().getRepositoryPolicy(getRepositoryPolicyRequest);
        response.whenComplete((resp, ex) -> {
            if (resp != null) {
                System.out.println("Repository policy retrieved successfully.");
            } else {
                if (ex.getCause() instanceof EcrException) {
                    throw (EcrException) ex.getCause();
                } else {
                    String errorMessage = "Unexpected error occurred: " +
ex.getMessage();
                    throw new RuntimeException(errorMessage, ex);
                }
            }
```

```
        });

        GetRepositoryPolicyResponse result = response.join();
        return result != null ? result.policyText() : null;
    }

    /**
     * Sets the repository policy for the specified ECR repository.
     *
     * @param repoName the name of the ECR repository.
     * @param iamRole  the IAM role to be granted access to the repository.
     * @throws RepositoryPolicyNotFoundException if the repository policy does not
exist.
     * @throws EcrException                      if there is an unexpected error
setting the repository policy.
     */
    public void setRepoPolicy(String repoName, String iamRole) {
        /*
          This example policy document grants the specified AWS principal the
permission to perform the
          `ecr:BatchGetImage` action. This policy is designed to allow the specified
principal
          to retrieve Docker images from the ECR repository.
         */
        String policyDocumentTemplate = """
             {
              "Version" : "2012-10-17",
              "Statement" : [ {
                "Sid" : "new statement",
                "Effect" : "Allow",
                "Principal" : {
                  "AWS" : "%s"
                },
                "Action" : "ecr:BatchGetImage"
              } ]
            }
             """;

        String policyDocument = String.format(policyDocumentTemplate, iamRole);
        SetRepositoryPolicyRequest setRepositoryPolicyRequest =
SetRepositoryPolicyRequest.builder()
            .repositoryName(repoName)
            .policyText(policyDocument)
            .build();
```

```
        CompletableFuture<SetRepositoryPolicyResponse> response =
getAsyncClient().setRepositoryPolicy(setRepositoryPolicyRequest);
        response.whenComplete((resp, ex) -> {
            if (resp != null) {
                System.out.println("Repository policy set successfully.");
            } else {
                Throwable cause = ex.getCause();
                if (cause instanceof RepositoryPolicyNotFoundException) {
                    throw (RepositoryPolicyNotFoundException) cause;
                } else if (cause instanceof EcrException) {
                    throw (EcrException) cause;
                } else {
                    String errorMessage = "Unexpected error: " + cause.getMessage();
                    throw new RuntimeException(errorMessage, cause);
                }
            }
        });
        response.join();
    }

    /**
     * Pushes a Docker image to an Amazon Elastic Container Registry (ECR)
repository.
     *
     * @param repoName  the name of the ECR repository to push the image to.
     * @param imageName the name of the Docker image.
     */
    public void pushDockerImage(String repoName, String imageName) {
        System.out.println("Pushing " + imageName + " to Amazon ECR will take a few
seconds.");
        CompletableFuture<AuthConfig> authResponseFuture =
getAsyncClient().getAuthorizationToken()
            .thenApply(response -> {
                String token =
response.authorizationData().get(0).authorizationToken();
                String decodedToken = new String(Base64.getDecoder().decode(token));
                String password = decodedToken.substring(4);

                DescribeRepositoriesResponse descrRepoResponse =
getAsyncClient().describeRepositories(b -> b.repositoryNames(repoName)).join();
                Repository repoData =
descrRepoResponse.repositories().stream().filter(r ->
r.repositoryName().equals(repoName)).findFirst().orElse(null);
```

```
                assert repoData != null;
                String registryURL = repoData.repositoryUri().split("/")[0];

                AuthConfig authConfig = new AuthConfig()
                    .withUsername("AWS")
                    .withPassword(password)
                    .withRegistryAddress(registryURL);
                return authConfig;
            })
            .thenCompose(authConfig -> {
                DescribeRepositoriesResponse descrRepoResponse =
 getAsyncClient().describeRepositories(b -> b.repositoryNames(repoName)).join();
                Repository repoData =
 descrRepoResponse.repositories().stream().filter(r ->
 r.repositoryName().equals(repoName)).findFirst().orElse(null);
                getDockerClient().tagImageCmd(imageName + ":latest",
 repoData.repositoryUri() + ":latest", imageName).exec();
                try {

 getDockerClient().pushImageCmd(repoData.repositoryUri()).withTag("echo-
text").withAuthConfig(authConfig).start().awaitCompletion();
                    System.out.println("The " + imageName + " was pushed to ECR");

                } catch (InterruptedException e) {
                    throw (RuntimeException) e.getCause();
                }
                return CompletableFuture.completedFuture(authConfig);
            });

        authResponseFuture.join();
    }

    // Make sure local image echo-text exists.
    public boolean isEchoTextImagePresent() {
        try {
            List<Image> images = getDockerClient().listImagesCmd().exec();
            boolean helloWorldFound = false;
            for (Image image : images) {
                String[] repoTags = image.getRepoTags();
                if (repoTags != null) {
                    for (String tag : repoTags) {
                        if (tag.startsWith("echo-text")) {
                            System.out.println(tag);
                            helloWorldFound = true;
```

```
                        }
                    }
                }
            }
            if (helloWorldFound) {
                System.out.println("The local image named echo-text exists.");
                return true;
            } else {
                System.out.println("The local image named echo-text does not
 exist.");
                return false;
            }
        } catch (DockerClientException ex) {
            logger.error("ERROR: " + ex.getMessage());
            return false;
        }
    }
}
```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella Documentazione di riferimento delle API AWS SDK for Java 2.x .
  - CreateRepository
  - DeleteRepository
  - DescribeImages
  - DescribeRepositories
  - GetAuthorizationToken
  - GetRepositoryPolicy
  - SetRepositoryPolicy
  - StartLifecyclePolicyPreview

Azioni

`CreateRepository`

Il seguente esempio di codice mostra come usare. `CreateRepository`

SDK per Java 2.x

> **Note**
>
> C'è 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 Amazon Elastic Container Registry (Amazon ECR) repository.
    *
    * @param repoName the name of the repository to create.
    * @return the Amazon Resource Name (ARN) of the created repository, or an empty
string if the operation failed.
    * @throws IllegalArgumentException     If repository name is invalid.
    * @throws RuntimeException             if an error occurs while creating the
repository.
    */
   public String createECRRepository(String repoName) {
       if (repoName == null || repoName.isEmpty()) {
           throw new IllegalArgumentException("Repository name cannot be null or
empty");
       }

       CreateRepositoryRequest request = CreateRepositoryRequest.builder()
           .repositoryName(repoName)
           .build();

       CompletableFuture<CreateRepositoryResponse> response =
getAsyncClient().createRepository(request);
       try {
           CreateRepositoryResponse result = response.join();
           if (result != null) {
               System.out.println("The " + repoName + " repository was created
successfully.");
               return result.repository().repositoryArn();
           } else {
               throw new RuntimeException("Unexpected response type");
           }
       } catch (CompletionException e) {
           Throwable cause = e.getCause();
           if (cause instanceof EcrException ex) {
```

```
                if
("RepositoryAlreadyExistsException".equals(ex.awsErrorDetails().errorCode())) {
                    System.out.println("The Amazon ECR repository already exists,
moving on...");
                    DescribeRepositoriesRequest describeRequest =
DescribeRepositoriesRequest.builder()
                        .repositoryNames(repoName)
                        .build();
                    DescribeRepositoriesResponse describeResponse =
getAsyncClient().describeRepositories(describeRequest).join();
                    return describeResponse.repositories().get(0).repositoryArn();
                } else {
                    throw new RuntimeException(ex);
                }
            } else {
                throw new RuntimeException(e);
            }
        }
    }
```

- Per i dettagli sull'API, CreateRepositoryconsulta AWS SDK for Java 2.x API Reference.

`DeleteRepository`

Il seguente esempio di codice mostra come utilizzare`DeleteRepository`.

SDK per Java 2.x

> **Note**
>
> C'è 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 ECR (Elastic Container Registry) repository.
     *
     * @param repoName the name of the repository to delete.
     * @throws IllegalArgumentException if the repository name is null or empty.
     * @throws EcrException if there is an error deleting the repository.
```

```
    * @throws RuntimeException if an unexpected error occurs during the deletion
process.
    */
   public void deleteECRRepository(String repoName) {
       if (repoName == null || repoName.isEmpty()) {
           throw new IllegalArgumentException("Repository name cannot be null or
empty");
       }

       DeleteRepositoryRequest repositoryRequest =
DeleteRepositoryRequest.builder()
           .force(true)
           .repositoryName(repoName)
           .build();

       CompletableFuture<DeleteRepositoryResponse> response =
getAsyncClient().deleteRepository(repositoryRequest);
       response.whenComplete((deleteRepositoryResponse, ex) -> {
           if (deleteRepositoryResponse != null) {
               System.out.println("You have successfully deleted the " + repoName +
" repository");
           } else {
               Throwable cause = ex.getCause();
               if (cause instanceof EcrException) {
                   throw (EcrException) cause;
               } else {
                   throw new RuntimeException("Unexpected error: " +
cause.getMessage(), cause);
               }
           }
       });

       // Wait for the CompletableFuture to complete
       response.join();
   }
```

- Per i dettagli sull'API, DeleteRepositoryconsulta AWS SDK for Java 2.x API Reference.

`DescribeImages`

Il seguente esempio di codice mostra come utilizzare`DescribeImages`.

SDK per Java 2.x

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
   /**
    * Verifies the existence of an image in an Amazon Elastic Container Registry
(Amazon ECR) repository asynchronously.
    *
    * @param repositoryName The name of the Amazon ECR repository.
    * @param imageTag       The tag of the image to verify.
    * @throws EcrException             if there is an error retrieving the image
information from Amazon ECR.
    * @throws CompletionException      if the asynchronous operation completes
exceptionally.
    */
   public void verifyImage(String repositoryName, String imageTag) {
       DescribeImagesRequest request = DescribeImagesRequest.builder()
           .repositoryName(repositoryName)
           .imageIds(ImageIdentifier.builder().imageTag(imageTag).build())
           .build();

       CompletableFuture<DescribeImagesResponse> response =
getAsyncClient().describeImages(request);
       response.whenComplete((describeImagesResponse, ex) -> {
           if (ex != null) {
               if (ex instanceof CompletionException) {
                   Throwable cause = ex.getCause();
                   if (cause instanceof EcrException) {
                       throw (EcrException) cause;
                   } else {
                       throw new RuntimeException("Unexpected error: " +
cause.getMessage(), cause);
                   }
               } else {
                   throw new RuntimeException("Unexpected error: " +
ex.getCause());
               }
```

```
            } else if (describeImagesResponse != null && !
describeImagesResponse.imageDetails().isEmpty()) {
                System.out.println("Image is present in the repository.");
            } else {
                System.out.println("Image is not present in the repository.");
            }
        });

        // Wait for the CompletableFuture to complete.
        response.join();
    }
```

- Per i dettagli sull'API, DescribeImagesconsulta AWS SDK for Java 2.x API Reference.

`DescribeRepositories`

Il seguente esempio di codice mostra come utilizzare`DescribeRepositories`.

SDK per Java 2.x

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
    /**
     * Retrieves the repository URI for the specified repository name.
     *
     * @param repoName the name of the repository to retrieve the URI for.
     * @return the repository URI for the specified repository name.
     * @throws EcrException        if there is an error retrieving the repository
information.
     * @throws CompletionException if the asynchronous operation completes
exceptionally.
     */
    public void getRepositoryURI(String repoName) {
        DescribeRepositoriesRequest request = DescribeRepositoriesRequest.builder()
            .repositoryNames(repoName)
            .build();
```

```
        CompletableFuture<DescribeRepositoriesResponse> response =
getAsyncClient().describeRepositories(request);
        response.whenComplete((describeRepositoriesResponse, ex) -> {
            if (ex != null) {
                Throwable cause = ex.getCause();
                if (cause instanceof InterruptedException) {
                    Thread.currentThread().interrupt();
                    String errorMessage = "Thread interrupted while waiting for
asynchronous operation: " + cause.getMessage();
                    throw new RuntimeException(errorMessage, cause);
                } else if (cause instanceof EcrException) {
                    throw (EcrException) cause;
                } else {
                    String errorMessage = "Unexpected error: " + cause.getMessage();
                    throw new RuntimeException(errorMessage, cause);
                }
            } else {
                if (describeRepositoriesResponse != null) {
                    if (!describeRepositoriesResponse.repositories().isEmpty()) {
                        String repositoryUri =
describeRepositoriesResponse.repositories().get(0).repositoryUri();
                        System.out.println("Repository URI found: " +
repositoryUri);
                    } else {
                        System.out.println("No repositories found for the given
name.");
                    }
                } else {
                    System.err.println("No response received from
describeRepositories.");
                }
            }
        });
        response.join();
    }
```

- Per i dettagli sull'API, DescribeRepositoriesconsulta AWS SDK for Java 2.x API Reference.

`GetAuthorizationToken`

Il seguente esempio di codice mostra come utilizzare`GetAuthorizationToken`.

SDK per Java 2.x

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
   /**
    * Retrieves the authorization token for Amazon Elastic Container Registry
(ECR).
    * This method makes an asynchronous call to the ECR client to retrieve the
authorization token.
    * If the operation is successful, the method prints the token to the console.
    * If an exception occurs, the method handles the exception and prints the error
message.
    *
    * @throws EcrException     if there is an error retrieving the authorization
token from ECR.
    * @throws RuntimeException if there is an unexpected error during the
operation.
    */
   public void getAuthToken() {
       CompletableFuture<GetAuthorizationTokenResponse> response =
getAsyncClient().getAuthorizationToken();
       response.whenComplete((authorizationTokenResponse, ex) -> {
           if (authorizationTokenResponse != null) {
               AuthorizationData authorizationData =
authorizationTokenResponse.authorizationData().get(0);
               String token = authorizationData.authorizationToken();
               if (!token.isEmpty()) {
                   System.out.println("The token was successfully retrieved.");
               }
           } else {
               if (ex.getCause() instanceof EcrException) {
                   throw (EcrException) ex.getCause();
               } else {
                   String errorMessage = "Unexpected error occurred: " +
ex.getMessage();
                   throw new RuntimeException(errorMessage, ex); // Rethrow the
exception
               }
```

```
            }
        });
        response.join();
    }
```

- Per i dettagli sull'API, GetAuthorizationTokenconsulta AWS SDK for Java 2.x API Reference.

`GetRepositoryPolicy`

Il seguente esempio di codice mostra come utilizzare`GetRepositoryPolicy`.

SDK per Java 2.x

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
    /**
     * Gets the repository policy for the specified repository.
     *
     * @param repoName the name of the repository.
     * @throws EcrException if an AWS error occurs while getting the repository
policy.
     */
    public String getRepoPolicy(String repoName) {
        if (repoName == null || repoName.isEmpty()) {
            throw new IllegalArgumentException("Repository name cannot be null or
empty");
        }

        GetRepositoryPolicyRequest getRepositoryPolicyRequest =
GetRepositoryPolicyRequest.builder()
            .repositoryName(repoName)
            .build();

        CompletableFuture<GetRepositoryPolicyResponse> response =
getAsyncClient().getRepositoryPolicy(getRepositoryPolicyRequest);
        response.whenComplete((resp, ex) -> {
            if (resp != null) {
```

```
                System.out.println("Repository policy retrieved successfully.");
            } else {
                if (ex.getCause() instanceof EcrException) {
                    throw (EcrException) ex.getCause();
                } else {
                    String errorMessage = "Unexpected error occurred: " +
ex.getMessage();
                    throw new RuntimeException(errorMessage, ex);
                }
            }
        });

        GetRepositoryPolicyResponse result = response.join();
        return result != null ? result.policyText() : null;
    }
```

- Per i dettagli sull'API, GetRepositoryPolicyconsulta AWS SDK for Java 2.x API Reference.

`PushImageCmd`

Il seguente esempio di codice mostra come utilizzare`PushImageCmd`.

SDK per Java 2.x

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
    /**
     * Pushes a Docker image to an Amazon Elastic Container Registry (ECR)
repository.
     *
     * @param repoName  the name of the ECR repository to push the image to.
     * @param imageName the name of the Docker image.
     */
    public void pushDockerImage(String repoName, String imageName) {
        System.out.println("Pushing " + imageName + " to Amazon ECR will take a few
seconds.");
```

```
        CompletableFuture<AuthConfig> authResponseFuture =
 getAsyncClient().getAuthorizationToken()
            .thenApply(response -> {
                String token =
 response.authorizationData().get(0).authorizationToken();
                String decodedToken = new String(Base64.getDecoder().decode(token));
                String password = decodedToken.substring(4);

                DescribeRepositoriesResponse descrRepoResponse =
 getAsyncClient().describeRepositories(b -> b.repositoryNames(repoName)).join();
                Repository repoData =
 descrRepoResponse.repositories().stream().filter(r ->
 r.repositoryName().equals(repoName)).findFirst().orElse(null);
                assert repoData != null;
                String registryURL = repoData.repositoryUri().split("/")[0];

                AuthConfig authConfig = new AuthConfig()
                    .withUsername("AWS")
                    .withPassword(password)
                    .withRegistryAddress(registryURL);
                return authConfig;
            })
            .thenCompose(authConfig -> {
                DescribeRepositoriesResponse descrRepoResponse =
 getAsyncClient().describeRepositories(b -> b.repositoryNames(repoName)).join();
                Repository repoData =
 descrRepoResponse.repositories().stream().filter(r ->
 r.repositoryName().equals(repoName)).findFirst().orElse(null);
                getDockerClient().tagImageCmd(imageName + ":latest",
 repoData.repositoryUri() + ":latest", imageName).exec();
                try {

 getDockerClient().pushImageCmd(repoData.repositoryUri()).withTag("echo-
text").withAuthConfig(authConfig).start().awaitCompletion();
                    System.out.println("The " + imageName + " was pushed to ECR");

                } catch (InterruptedException e) {
                    throw (RuntimeException) e.getCause();
                }
                return CompletableFuture.completedFuture(authConfig);
            });

        authResponseFuture.join();
    }
```

- Per i dettagli sull'API, PushImageCmdconsulta AWS SDK for Java 2.x API Reference.

`SetRepositoryPolicy`

Il seguente esempio di codice mostra come utilizzare`SetRepositoryPolicy`.

SDK per Java 2.x

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
   /**
    * Sets the repository policy for the specified ECR repository.
    *
    * @param repoName the name of the ECR repository.
    * @param iamRole  the IAM role to be granted access to the repository.
    * @throws RepositoryPolicyNotFoundException if the repository policy does not
exist.
    * @throws EcrException                      if there is an unexpected error
setting the repository policy.
    */
   public void setRepoPolicy(String repoName, String iamRole) {
       /*
         This example policy document grants the specified AWS principal the
permission to perform the
         `ecr:BatchGetImage` action. This policy is designed to allow the specified
principal
         to retrieve Docker images from the ECR repository.
        */
       String policyDocumentTemplate = """
           {
            "Version" : "2012-10-17",
            "Statement" : [ {
              "Sid" : "new statement",
              "Effect" : "Allow",
              "Principal" : {
```

```
                  "AWS" : "%s"
               },
               "Action" : "ecr:BatchGetImage"
             } ]
           }
            """;

       String policyDocument = String.format(policyDocumentTemplate, iamRole);
       SetRepositoryPolicyRequest setRepositoryPolicyRequest =
SetRepositoryPolicyRequest.builder()
           .repositoryName(repoName)
           .policyText(policyDocument)
           .build();

       CompletableFuture<SetRepositoryPolicyResponse> response =
getAsyncClient().setRepositoryPolicy(setRepositoryPolicyRequest);
       response.whenComplete((resp, ex) -> {
           if (resp != null) {
               System.out.println("Repository policy set successfully.");
           } else {
               Throwable cause = ex.getCause();
               if (cause instanceof RepositoryPolicyNotFoundException) {
                   throw (RepositoryPolicyNotFoundException) cause;
               } else if (cause instanceof EcrException) {
                   throw (EcrException) cause;
               } else {
                   String errorMessage = "Unexpected error: " + cause.getMessage();
                   throw new RuntimeException(errorMessage, cause);
               }
           }
       });
       response.join();
   }
```

- Per i dettagli sull'API, SetRepositoryPolicyconsulta AWS SDK for Java 2.x API Reference.

`StartLifecyclePolicyPreview`

Il seguente esempio di codice mostra come utilizzare`StartLifecyclePolicyPreview`.

SDK per Java 2.x

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
   /**
    * Verifies the existence of an image in an Amazon Elastic Container Registry
(Amazon ECR) repository asynchronously.
    *
    * @param repositoryName The name of the Amazon ECR repository.
    * @param imageTag       The tag of the image to verify.
    * @throws EcrException             if there is an error retrieving the image
information from Amazon ECR.
    * @throws CompletionException      if the asynchronous operation completes
exceptionally.
    */
   public void verifyImage(String repositoryName, String imageTag) {
       DescribeImagesRequest request = DescribeImagesRequest.builder()
           .repositoryName(repositoryName)
           .imageIds(ImageIdentifier.builder().imageTag(imageTag).build())
           .build();

       CompletableFuture<DescribeImagesResponse> response =
getAsyncClient().describeImages(request);
       response.whenComplete((describeImagesResponse, ex) -> {
           if (ex != null) {
               if (ex instanceof CompletionException) {
                   Throwable cause = ex.getCause();
                   if (cause instanceof EcrException) {
                       throw (EcrException) cause;
                   } else {
                       throw new RuntimeException("Unexpected error: " +
cause.getMessage(), cause);
                   }
               } else {
                   throw new RuntimeException("Unexpected error: " +
ex.getCause());
               }
```

```
            } else if (describeImagesResponse != null && !
describeImagesResponse.imageDetails().isEmpty()) {
                System.out.println("Image is present in the repository.");
            } else {
                System.out.println("Image is not present in the repository.");
            }
        });

        // Wait for the CompletableFuture to complete.
        response.join();
    }
```

- Per i dettagli sull'API, StartLifecyclePolicyPreviewconsulta AWS SDK for Java 2.x API Reference.

Esempi di Amazon ECS che utilizzano 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.

Ogni esempio include un collegamento al codice sorgente completo, 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, CreateClusterconsulta 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, CreateServiceconsulta 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, DeleteServiceconsulta 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, DescribeClustersconsulta 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, DescribeTasksconsulta 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, ListClustersconsulta 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, UpdateServiceconsulta 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.

Gli scenari sono esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio o combinate con altri Servizi AWS.

Ogni esempio include un collegamento al codice sorgente completo, in cui 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, DescribeLoadBalancersconsulta 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)
                    .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, CreateListenerconsulta 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)
                    .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, CreateLoadBalancerconsulta 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, CreateTargetGroupconsulta 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, DeleteLoadBalancerconsulta 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, DeleteTargetGroupconsulta 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, DescribeTargetHealthconsulta 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 Amazon EC2 Auto Scaling per creare istanze Amazon Elastic Compute Cloud (Amazon EC2) basate su un modello di avvio e per mantenere il numero di istanze in un intervallo specificato.
- Gestisci e distribuisci 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 EC2 istanza 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 di integrità aggiornando AWS Systems Manager i parametri.

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.

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 wasSuccessul = loadBalancer.verifyLoadBalancerEndpoint(elbDnsName);
        if (!wasSuccessul) {
            System.out.println("Couldn't connect to the load balancer, verifying
that the port is open...");
            CloseableHttpClient httpClient = HttpClients.createDefault();

            // Create an HTTP GET request to "http://checkip.amazonaws.com"
            HttpGet httpGet = new HttpGet("http://checkip.amazonaws.com");
            try {
```

```
                // Execute the request and get the response
                HttpResponse response = httpClient.execute(httpGet);

                // Read the response content.
                String ipAddress =
IOUtils.toString(response.getEntity().getContent(), StandardCharsets.UTF_8).trim();

                // Print the public IP address.
                System.out.println("Public IP Address: " + ipAddress);
                GroupInfo groupInfo = autoScaler.verifyInboundPort(vpcId, port,
ipAddress);
                if (!groupInfo.isPortOpen()) {
                    System.out.println("""
                            For this example to work, the default security group for
your default VPC must
                            allow access from this computer. You can either add it
automatically from this
                            example or add it yourself using the AWS Management
Console.
                            """);

                    System.out.println(
                            "Do you want to add a rule to security group " +
groupInfo.getGroupName() + " to allow");
                    System.out.println("inbound traffic on port " + port + " from
your computer's IP address (y/n) ");
                    String ans = in.nextLine();
                    if ("y".equalsIgnoreCase(ans)) {
                        autoScaler.openInboundPort(groupInfo.getGroupName(),
String.valueOf(port), ipAddress);
                        System.out.println("Security group rule added.");
                    } else {
                        System.out.println("No security group rule added.");
                    }
                }

            } catch (AutoScalingException e) {
                e.printStackTrace();
            }
        } else if (wasSuccessul) {
            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 azioni di Auto Scaling e EC2 Amazon.

```
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.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)
                    .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.

Ogni esempio include un collegamento al codice sorgente completo, in cui 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, CreateContainerconsulta 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, DeleteContainerconsulta 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, DeleteObjectconsulta 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, DescribeContainerconsulta 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 outStream = new FileOutputStream(targetFile);
            outStream.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, GetObjectconsulta 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, ListContainersconsulta 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
 to use, and path in the container\s

                Ex: <containerName> <filePath> <completePath>
                """;

        if (args.length < 3) {
            System.out.println(USAGE);
            System.exit(1);
        }

        String containerName = args[0];
        String filePath = args[1];
        String completePath = args[2];

        Region region = Region.US_EAST_1;
        URI uri = new URI(getEndpoint(containerName));
        MediaStoreDataClient mediaStoreData = MediaStoreDataClient.builder()
                .endpointOverride(uri)
                .region(region)
                .build();

        putMediaObject(mediaStoreData, filePath, completePath);
        mediaStoreData.close();
    }

    public static void putMediaObject(MediaStoreDataClient mediaStoreData, String
 filePath, String completePath) {
        try {
            File myFile = new File(filePath);
            RequestBody requestBody = RequestBody.fromFile(myFile);
```

```
            PutObjectRequest objectRequest = PutObjectRequest.builder()
                    .path(completePath)
                    .contentType("video/mp4")
                    .build();

            PutObjectResponse response = mediaStoreData.putObject(objectRequest,
 requestBody);
            System.out.println("The saved object is " +
 response.storageClass().toString());

        } catch (MediaStoreDataException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static String getEndpoint(String containerName) {

        Region region = Region.US_EAST_1;
        MediaStoreClient mediaStoreClient = MediaStoreClient.builder()
                .region(region)
                .build();

        DescribeContainerRequest containerRequest =
 DescribeContainerRequest.builder()
                .containerName(containerName)
                .build();

        DescribeContainerResponse response =
 mediaStoreClient.describeContainer(containerRequest);
        return response.container().endpoint();
    }
}
```

- Per i dettagli sull'API, PutObjectconsulta AWS SDK for Java 2.x API Reference.

AWS Entity Resolution 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 Entity Resolution.

Le nozioni di base sono esempi di codice che mostrano come eseguire le operazioni essenziali all'interno di un servizio.

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.

Ogni esempio include un collegamento al codice sorgente completo, in cui è possibile trovare istruzioni su come configurare ed eseguire il codice nel contesto.

Nozioni di base

Salve AWS Entity Resolution

L'esempio di codice seguente mostra come iniziare a utilizzare AWS Entity Resolution.

SDK per Java 2.x

> **Note**
>
> C'è 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
 */
public class HelloEntityResoultion {

    private static final Logger logger =
 LoggerFactory.getLogger(HelloEntityResoultion.class);

    private static EntityResolutionAsyncClient entityResolutionAsyncClient;
    public static void main(String[] args) {
        listMatchingWorkflows();
    }
```

```
   public static EntityResolutionAsyncClient getResolutionAsyncClient() {
       if (entityResolutionAsyncClient == null) {
           /*
           The `NettyNioAsyncHttpClient` class is part of the AWS SDK for Java,
version 2,
           and it is designed to provide a high-performance, asynchronous HTTP
client for interacting with AWS services.
            It uses the Netty framework to handle the underlying network
communication and the Java NIO API to
            provide a non-blocking, event-driven approach to HTTP requests and
responses.
            */

           SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
               .maxConcurrency(50)  // Adjust as needed.
               .connectionTimeout(Duration.ofSeconds(60))  // Set the connection
timeout.
               .readTimeout(Duration.ofSeconds(60))  // Set the read timeout.
               .writeTimeout(Duration.ofSeconds(60))  // Set the write timeout.
               .build();

           ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
               .apiCallTimeout(Duration.ofMinutes(2))  // Set the overall API call
timeout.
               .apiCallAttemptTimeout(Duration.ofSeconds(90))  // Set the
individual call attempt timeout.
               .retryStrategy(RetryMode.STANDARD)
               .build();

           entityResolutionAsyncClient = EntityResolutionAsyncClient.builder()
               .httpClient(httpClient)
               .overrideConfiguration(overrideConfig)
               .build();
       }
       return entityResolutionAsyncClient;
   }

   /**
    * Lists all matching workflows using an asynchronous paginator.
    * <p>
    * This method requests a paginated list of matching workflows from the
    * AWS Entity Resolution service and logs the names of the retrieved workflows.
    * It uses an asynchronous approach with a paginator and waits for the operation
```

```
     * to complete using {@code CompletableFuture#join()}.
     * </p>
     */
    public static void listMatchingWorkflows() {
        ListMatchingWorkflowsRequest request =
 ListMatchingWorkflowsRequest.builder().build();

        ListMatchingWorkflowsPublisher paginator =
            getResolutionAsyncClient().listMatchingWorkflowsPaginator(request);

        // Iterate through the paginated results asynchronously
        CompletableFuture<Void> future = paginator.subscribe(response -> {
            response.workflowSummaries().forEach(workflow ->
                logger.info("Matching Workflow Name: " + workflow.workflowName())
            );
        });

        // Wait for the asynchronous operation to complete
        future.join();
    }
}
```

- Per i dettagli sull'API, listMatchingWorkflowsconsulta AWS SDK for Java 2.x API Reference.

Argomenti

- Nozioni di base
- Azioni

Nozioni di base

Informazioni di base

L'esempio di codice seguente mostra come:

- Crea la mappatura dello schema.
- Crea un AWS Entity Resolution flusso di lavoro.
- Avvia il processo corrispondente per il flusso di lavoro.
- Ottieni i dettagli sul lavoro corrispondente.

- Ottieni la mappatura dello schema.
- Elenca tutte le mappature dello schema.
- Etichetta la risorsa Schema Mapping.
- Eliminare gli AWS Entity Resolution asset.

SDK per Java 2.x

> **Note**
>
> C'è altro da fare 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 AWS Entity Resolution funzionalità.

```
public class EntityResScenario {
    private static final Logger logger =
 LoggerFactory.getLogger(EntityResScenario.class);
    public static final String DASHES = new String(new char[80]).replace("\0", "-");
    private static final String STACK_NAME = "EntityResolutionCdkStack";
    private static final String ENTITY_RESOLUTION_ROLE_ARN_KEY =
 "EntityResolutionRoleArn";
    private static final String GLUE_DATA_BUCKET_NAME_KEY = "GlueDataBucketName";
    private static final String JSON_GLUE_TABLE_ARN_KEY = "JsonErGlueTableArn";
    private static final String CSV_GLUE_TABLE_ARN_KEY = "CsvErGlueTableArn";
    private static String glueBucketName;
    private static String workflowName = "workflow-" + UUID.randomUUID();

    private static String jsonSchemaMappingName = "jsonschema-" + UUID.randomUUID();
    private static String jsonSchemaMappingArn = null;
    private static String csvSchemaMappingName = "csv-" + UUID.randomUUID();
    private static String roleARN;
    private static String csvGlueTableArn;
    private static String jsonGlueTableArn;
    private static Scanner scanner = new Scanner(System.in);

    private static EntityResActions actions = new EntityResActions();

    public static void main(String[] args) throws InterruptedException {

        logger.info("Welcome to the AWS Entity Resolution Scenario.");
```

```
        logger.info("""
            AWS Entity Resolution is a fully-managed machine learning service
provided by
            Amazon Web Services (AWS) that helps organizations extract, link, and
            organize information from multiple data sources. It leverages natural
            language processing and deep learning models to identify and resolve
            entities, such as people, places, organizations, and products,
            across structured and unstructured data.

            With Entity Resolution, customers can build robust data integration
            pipelines to combine and reconcile data from multiple systems,
databases,
            and documents. The service can handle ambiguous, incomplete, or
conflicting
            information, and provide a unified view of entities and their
relationships.
            This can be particularly valuable in applications such as customer 360,
            fraud detection, supply chain management, and knowledge management,
where
            accurate entity identification is crucial.

            The `EntityResolutionAsyncClient` interface in the AWS SDK for Java 2.x
            provides a set of methods to programmatically interact with the AWS
Entity
            Resolution service. This allows developers to automate the entity
extraction,
            linking, and deduplication process as part of their data processing
workflows.
            With Entity Resolution, organizations can unlock the value of their
data,
            improve decision-making, and enhance customer experiences by having a
reliable,
            comprehensive view of their key entities.
            """);

        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("""
            To prepare the AWS resources needed for this scenario application, the
next step uploads
            a CloudFormation template whose resulting stack creates the following
resources:
```

```
            - An AWS Glue Data Catalog table
            - An AWS IAM role
            - An AWS S3 bucket
            - An AWS Entity Resolution Schema

            It can take a couple minutes for the Stack to finish creating the
resources.
            """);
        waitForInputToContinue(scanner);
        logger.info("Generating resources...");
        CloudFormationHelper.deployCloudFormationStack(STACK_NAME);
        Map<String, String> outputsMap =
CloudFormationHelper.getStackOutputsAsync(STACK_NAME).join();
        roleARN = outputsMap.get(ENTITY_RESOLUTION_ROLE_ARN_KEY);
        glueBucketName = outputsMap.get(GLUE_DATA_BUCKET_NAME_KEY);
        csvGlueTableArn = outputsMap.get(CSV_GLUE_TABLE_ARN_KEY);
        jsonGlueTableArn = outputsMap.get(JSON_GLUE_TABLE_ARN_KEY);
        logger.info(DASHES);
        waitForInputToContinue(scanner);

        try {
            runScenario();

        } catch (Exception ce) {
            Throwable cause = ce.getCause();
            logger.error("An exception happened: " + (cause != null ?
cause.getMessage() : ce.getMessage()));
        }
    }

    private static void runScenario() throws InterruptedException {
        /*
         This JSON is a valid input for the AWS Entity Resolution service.
         The JSON represents an array of three objects, each containing an "id",
"name", and "email"
         property. This format aligns with the expected input structure for the
         Entity Resolution service.
         */
        String json = """
            {"id":"1","name":"Jane Doe","email":"jane.doe@example.com"}
            {"id":"2","name":"John Doe","email":"john.doe@example.com"}
            {"id":"3","name":"Jorge Souza","email":"jorge_souza@example.com"}
            """;
```

```
        logger.info("Upload the following JSON objects to the {} S3 bucket.",
 glueBucketName);
        logger.info(json);
        String csv = """
            id,name,email,phone
            1,Jane B.,Doe,jane.doe@example.com,555-876-9846
            2,John Doe Jr.,john.doe@example.com,555-654-3210
            3,María García,maría_garcia@company.com,555-567-1234
            4,Mary Major,mary_major@company.com,555-222-3333
            """;
        logger.info("Upload the following CSV data to the {} S3 bucket.",
 glueBucketName);
        logger.info(csv);
        waitForInputToContinue(scanner);
        try {
            actions.uploadInputData(glueBucketName, json, csv);
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();

            if (cause == null) {
                logger.error("Failed to upload input data: {}", ce.getMessage(),
 ce);
            }

            if (cause instanceof ResourceNotFoundException) {
                logger.error("Failed to upload input data as the resource was not
 found: {}", cause.getMessage(), cause);
            }
            return;
        }
        logger.info("The JSON and CSV objects have been uploaded to the S3
 bucket.");
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("1. Create Schema Mapping");
        logger.info("""
            Entity Resolution schema mapping aligns and integrates data from
            multiple sources by identifying and matching corresponding entities
            like customers or products. It unifies schemas, resolves conflicts,
            and uses machine learning to link related entities, enabling a
            consolidated, accurate view for improved data quality and decision-
making.
```

```
            In this example, the schema mapping lines up with the fields in the JSON
and CSV objects. That is,
            it contains these fields: id, name, and email.
            """);
        try {
            CreateSchemaMappingResponse response =
actions.createSchemaMappingAsync(jsonSchemaMappingName).join();
            jsonSchemaMappingName = response.schemaName();
            logger.info("The JSON schema mapping name is " + jsonSchemaMappingName);
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();

            if (cause == null) {
                logger.error("Failed to create JSON schema mapping: {}",
ce.getMessage(), ce);
            }

            if (cause instanceof ConflictException) {
                logger.error("Schema mapping conflict detected: {}",
cause.getMessage(), cause);
            } else {
                logger.error("Unexpected error while creating schema mapping: {}",
cause.getMessage(), cause);
            }
            return;
        }

        try {
            CreateSchemaMappingResponse response =
actions.createSchemaMappingAsync(csvSchemaMappingName).join();
            csvSchemaMappingName = response.schemaName();
            logger.info("The CSV schema mapping name is " + csvSchemaMappingName);
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause == null) {
                logger.error("Failed to create CSV schema mapping: {}",
ce.getMessage(), ce);
            }

            if (cause instanceof ConflictException) {
                logger.error("Schema mapping conflict detected: {}",
cause.getMessage(), cause);
            } else {
```

```
                logger.error("Unexpected error while creating CSV schema mapping:
{}", cause.getMessage(), cause);
            }
            return;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("2. Create an AWS Entity Resolution Workflow. ");
        logger.info("""
            An Entity Resolution matching workflow identifies and links records
            across datasets that represent the same real-world entity, such as
            customers or products. Using techniques like schema mapping,
            data profiling, and machine learning algorithms,
            it evaluates attributes like names or emails to detect duplicates
            or relationships, even with variations or inconsistencies.
            The workflow outputs consolidated, de-duplicated data.

            We will use the machine learning-based matching technique.
            """);
        waitForInputToContinue(scanner);
        try {
            String workflowArn = actions.createMatchingWorkflowAsync(
                roleARN, workflowName, glueBucketName, jsonGlueTableArn,
                jsonSchemaMappingName, csvGlueTableArn,
csvSchemaMappingName).join();

            logger.info("The workflow ARN is: " + workflowArn);
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();

            if (cause == null) {
                logger.error("An unexpected error occurred: {}", ce.getMessage(),
ce);
            }

            if (cause instanceof ValidationException) {
                logger.error("Validation error: {}", cause.getMessage(), cause);
            } else if (cause instanceof ConflictException) {
                logger.error("Workflow conflict detected: {}", cause.getMessage(),
cause);
            } else {
                logger.error("Unexpected error: {}", cause.getMessage(), cause);
```

```
            }
            return;
        }

        waitForInputToContinue(scanner);
        logger.info(DASHES);
        logger.info("3. Start the matching job of the " + workflowName + "
workflow.");
        waitForInputToContinue(scanner);
        String jobId = null;
        try {
            jobId = actions.startMatchingJobAsync(workflowName).join();
            logger.info("The matching job was successfully started.");
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ConflictException) {
                logger.error("Job conflict detected: {}", cause.getMessage(),
cause);
            } else {
                logger.error("Unexpected error while starting the job: {}",
ce.getMessage(), ce);
            }
            return;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("4. While the matching job is running, let's look at other API
methods. First, let's get details for job " + jobId);
        waitForInputToContinue(scanner);
        try {
            actions.getMatchingJobAsync(jobId, workflowName).join();
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ResourceNotFoundException) {
                logger.error("The matching job not found: {}", cause.getMessage(),
cause);
            } else {
                logger.error("Failed to start matching job: " + (cause != null ?
cause.getMessage() : ce.getMessage()));
            }
            return;
        }
```

```
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("5. Get the schema mapping for the JSON data.");
        waitForInputToContinue(scanner);
        try {
            GetSchemaMappingResponse response =
actions.getSchemaMappingAsync(jsonSchemaMappingName).join();
            jsonSchemaMappingArn = response.schemaArn();
            logger.info("Schema mapping ARN is " + jsonSchemaMappingArn);
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ResourceNotFoundException) {
                logger.error("Schema mapping not found: {}", cause.getMessage(),
cause);
            } else {
                logger.error("Error retrieving the specific schema mapping: " +
ce.getCause().getMessage());
            }
            return;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("6. List Schema Mappings.");
        try {
            actions.ListSchemaMappings();
        } catch (CompletionException ce) {
            logger.error("Error retrieving schema mappings: " +
ce.getCause().getMessage());
            return;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("7. Tag the {} resource.", jsonSchemaMappingName);
        logger.info("""
            Tags can help you organize and categorize your Entity Resolution
resources.
            You can also use them to scope user permissions by granting a user
permission
            to access or change only resources with certain tag values.
```

```
            In Entity Resolution, SchemaMapping and MatchingWorkflow can be tagged.
For this example,
            the SchemaMapping is tagged.
                """);
        try {
            actions.tagEntityResource(jsonSchemaMappingArn).join();
        } catch (CompletionException ce) {
            logger.error("Error tagging the resource: " +
ce.getCause().getMessage());
            return;
        }

        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("8. View the results of the AWS Entity Resolution Workflow.");
        logger.info("""
            You cannot view the result of the workflow that is in a running state.
            In order to view the results, you need to wait for the workflow that we
started in step 3 to complete.

            If you choose not to wait, you cannot view the results. You can perform

            this task manually in the AWS Management Console.

            This can take up to 30 mins (y/n).
            """);
        String viewAns = scanner.nextLine().trim();
        boolean isComplete = false;
        if (viewAns.equalsIgnoreCase("y")) {
            logger.info("You selected to view the Entity Resolution Workflow
results.");
            countdownWithWorkflowCheck(actions, 1800, jobId, workflowName);
            isComplete = true;
            try {
                JobMetrics metrics = actions.getJobInfo(workflowName, jobId).join();
                logger.info("Number of input records: {}", metrics.inputRecords());
                logger.info("Number of match ids: {}", metrics.matchIDs());
                logger.info("Number of records not processed: {}",
metrics.recordsNotProcessed());
                logger.info("Number of total records processed: {}",
metrics.totalRecordsProcessed());
```

```
                logger.info("The following represents the output data generated by
the Entity Resolution workflow based on the JSON and CSV input data. The output
data is stored in the {} bucket.", glueBucketName);
                actions.printData(glueBucketName);

                logger.info("""

                    Note that each of the last 2 records are considered a match even
though the 'name' differs between the records;
                    For example 'John Doe Jr.' compared to 'John Doe'.
                    The confidence level is a value between 0 and 1, where 1
indicates a perfect match.

                    """);

            } catch (CompletionException ce) {
                Throwable cause = ce.getCause();
                if (cause instanceof ResourceNotFoundException) {
                    logger.error("The job not found: {}", cause.getMessage(),
cause);
                } else {
                    logger.error("Error retrieving job information: " +
ce.getCause().getMessage());
                }
                return;
            }
        }

        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("9. Do you want to delete the resources, including the workflow?
(y/n)");
        logger.info("""
            You cannot delete the workflow that is in a running state.
            In order to delete the workflow, you need to wait for the workflow to
complete.

            You can delete the workflow manually in the AWS Management Console at a
later time.

            If you already waited for the workflow to complete in the previous
step,
```

```
            the workflow is completed and you can delete it.

            If the workflow is not completed, this can take up to 30 mins (y/n).
            """);
        String delAns = scanner.nextLine().trim();
        if (delAns.equalsIgnoreCase("y")) {
            try {
                if (!isComplete) {
                    countdownWithWorkflowCheck(actions, 1800, jobId, workflowName);
                }
                actions.deleteMatchingWorkflowAsync(workflowName).join();
                logger.info("Workflow deleted successfully!");
            } catch (CompletionException ce) {
                logger.info("Error deleting the workflow: {} ", ce.getMessage());
                return;
            }

            try {
                // Delete both schema mappings.
                actions.deleteSchemaMappingAsync(jsonSchemaMappingName).join();
                actions.deleteSchemaMappingAsync(csvSchemaMappingName).join();
                logger.info("Both schema mappings were deleted successfully!");
            } catch (CompletionException ce) {
                logger.error("Error deleting schema mapping: {}", ce.getMessage());
                return;
            }

            waitForInputToContinue(scanner);
            logger.info(DASHES);
            logger.info("""
                Now we delete the CloudFormation stack, which deletes
                the resources that were created at the beginning of this scenario.
                """);
            waitForInputToContinue(scanner);
            logger.info(DASHES);
            try {
                deleteCloudFormationStack();
            } catch (RuntimeException e) {
                logger.error("Failed to delete the stack: {}", e.getMessage());
                return;
            }

        } else {
```

```
            logger.info("You can delete the AWS resources in the AWS Management
Console.");
        }

        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("This concludes the AWS Entity Resolution scenario.");
        logger.info(DASHES);
    }

    private static void waitForInputToContinue(Scanner scanner) {
        while (true) {
            logger.info("");
            logger.info("Enter 'c' followed by <ENTER> to continue:");
            String input = scanner.nextLine();

            if (input.trim().equalsIgnoreCase("c")) {
                logger.info("Continuing with the program...");
                logger.info("");
                break;
            } else {
                // Handle invalid input.
                logger.info("Invalid input. Please try again.");
            }
        }
    }

    public static void countdownWithWorkflowCheck(EntityResActions actions, int
totalSeconds, String jobId, String workflowName) throws InterruptedException {
        int secondsElapsed = 0;

        while (true) {
            // Calculate display minutes and seconds.
            int remainingTime = totalSeconds - secondsElapsed;
            int displayMinutes = remainingTime / 60;
            int displaySeconds = remainingTime % 60;

            // Print the countdown.
            System.out.printf("\r%02d:%02d", displayMinutes, displaySeconds);
            Thread.sleep(1000); // Wait for 1 second
            secondsElapsed++;
```

```
            // Check workflow status every 60 seconds.
            if (secondsElapsed % 60 == 0 || remainingTime <= 0) {
                GetMatchingJobResponse response =
 actions.checkWorkflowStatusCompleteAsync(jobId, workflowName).join();
                if (response != null &&
 "SUCCEEDED".equalsIgnoreCase(String.valueOf(response.status()))) {
                    logger.info(""); // Move to the next line after countdown.
                    logger.info("Countdown complete: Workflow is in Completed
 state!");
                    break; // Break out of the loop if the status is "SUCCEEDED"
                }
            }

            // If countdown reaches zero, reset it for continuous countdown.
            if (remainingTime <= 0) {
                secondsElapsed = 0;
            }
        }
    }

    private static void deleteCloudFormationStack() {
        try {
            CloudFormationHelper.emptyS3Bucket(glueBucketName);
            CloudFormationHelper.destroyCloudFormationStack(STACK_NAME);
            logger.info("Resources deleted successfully!");
        } catch (CloudFormationException e) {
            throw new RuntimeException("Failed to delete CloudFormation stack: " +
 e.getMessage(), e);
        } catch (S3Exception e) {
            throw new RuntimeException("Failed to empty S3 bucket: " +
 e.getMessage(), e);
        }
    }
}
```

Una classe wrapper per i metodi SDK. AWS Entity Resolution

```
public class EntityResActions {

    private static final String PREFIX = "eroutput/";
    private static final Logger logger =
 LoggerFactory.getLogger(EntityResActions.class);
```

```
   private static EntityResolutionAsyncClient entityResolutionAsyncClient;

   private static S3AsyncClient s3AsyncClient;

   public static EntityResolutionAsyncClient getResolutionAsyncClient() {
       if (entityResolutionAsyncClient == null) {
           /*
           The `NettyNioAsyncHttpClient` class is part of the AWS SDK for Java,
version 2,
           and it is designed to provide a high-performance, asynchronous HTTP
client for interacting with AWS services.
            It uses the Netty framework to handle the underlying network
communication and the Java NIO API to
            provide a non-blocking, event-driven approach to HTTP requests and
responses.
            */

           SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
               .maxConcurrency(50)  // Adjust as needed.
               .connectionTimeout(Duration.ofSeconds(60))  // Set the connection
timeout.
               .readTimeout(Duration.ofSeconds(60))  // Set the read timeout.
               .writeTimeout(Duration.ofSeconds(60))  // Set the write timeout.
               .build();

           ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
               .apiCallTimeout(Duration.ofMinutes(2))  // Set the overall API call
timeout.
               .apiCallAttemptTimeout(Duration.ofSeconds(90))  // Set the
individual call attempt timeout.
               .retryStrategy(RetryMode.STANDARD)
               .build();

           entityResolutionAsyncClient = EntityResolutionAsyncClient.builder()
               .httpClient(httpClient)
               .overrideConfiguration(overrideConfig)
               .build();
       }
       return entityResolutionAsyncClient;
   }

   public static S3AsyncClient getS3AsyncClient() {
```

```
        if (s3AsyncClient == null) {
            /*
            The `NettyNioAsyncHttpClient` class is part of the AWS SDK for Java,
version 2,
            and it is designed to provide a high-performance, asynchronous HTTP
client for interacting with AWS services.
             It uses the Netty framework to handle the underlying network
communication and the Java NIO API to
             provide a non-blocking, event-driven approach to HTTP requests and
responses.
             */

            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                .maxConcurrency(50)  // Adjust as needed.
                .connectionTimeout(Duration.ofSeconds(60))  // Set the connection
timeout.
                .readTimeout(Duration.ofSeconds(60))  // Set the read timeout.
                .writeTimeout(Duration.ofSeconds(60))  // Set the write timeout.
                .build();

            ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
                .apiCallTimeout(Duration.ofMinutes(2))  // Set the overall API call
timeout.
                .apiCallAttemptTimeout(Duration.ofSeconds(90))  // Set the
individual call attempt timeout.
                .retryStrategy(RetryMode.STANDARD)
                .build();

            s3AsyncClient = S3AsyncClient.builder()
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return s3AsyncClient;
    }

    /**
     * Deletes the schema mapping asynchronously.
     *
     * @param schemaName the name of the schema to delete
     * @return a {@link CompletableFuture} that completes when the schema mapping is
deleted successfully,
     * or throws a {@link RuntimeException} if the deletion fails
```

```
     */
    public CompletableFuture<DeleteSchemaMappingResponse>
deleteSchemaMappingAsync(String schemaName) {
        DeleteSchemaMappingRequest request = DeleteSchemaMappingRequest.builder()
            .schemaName(schemaName)
            .build();

        return getResolutionAsyncClient().deleteSchemaMapping(request)
            .whenComplete((response, exception) -> {
                if (response != null) {
                    // Successfully deleted the schema mapping, log the success
message.
                    logger.info("Schema mapping '{}' deleted successfully.",
schemaName);
                } else {
                    // Ensure exception is not null before accessing its cause.
                    if (exception == null) {
                        throw new CompletionException("An unknown error occurred
while deleting the schema mapping.", null);
                    }

                    Throwable cause = exception.getCause();
                    if (cause instanceof ResourceNotFoundException) {
                        throw new CompletionException("The schema mapping was not
found to delete: " + schemaName, cause);
                    }

                    // Wrap other AWS exceptions in a CompletionException.
                    throw new CompletionException("Failed to delete schema mapping:
" + schemaName, exception);
                }
            });
    }

    /**
     * Lists the schema mappings associated with the current AWS account. This
method uses an asynchronous paginator to
     * retrieve the schema mappings, and prints the name of each schema mapping to
the console.
     */
    public void ListSchemaMappings() {
        ListSchemaMappingsRequest mappingsRequest =
ListSchemaMappingsRequest.builder()
            .build();
```

```
        ListSchemaMappingsPublisher paginator =
getResolutionAsyncClient().listSchemaMappingsPaginator(mappingsRequest);

        // Iterate through the pages of results
        CompletableFuture<Void> future = paginator.subscribe(response -> {
            response.schemaList().forEach(schemaMapping ->
                logger.info("Schema Mapping Name: " + schemaMapping.schemaName())
            );
        });

        // Wait for the asynchronous operation to complete
        future.join();
    }

    /**
     * Asynchronously deletes a workflow with the specified name.
     *
     * @param workflowName the name of the workflow to be deleted
     * @return a {@link CompletableFuture} that completes when the workflow has been
deleted
     * @throws RuntimeException if the deletion of the workflow fails
     */
    public CompletableFuture<DeleteMatchingWorkflowResponse>
deleteMatchingWorkflowAsync(String workflowName) {
        DeleteMatchingWorkflowRequest request =
DeleteMatchingWorkflowRequest.builder()
            .workflowName(workflowName)
            .build();

        return getResolutionAsyncClient().deleteMatchingWorkflow(request)
            .whenComplete((response, exception) -> {
                if (response != null) {
                    logger.info("{} was deleted", workflowName );
                } else {
                    if (exception == null) {
                        throw new CompletionException("An unknown error occurred
while deleting the workflow.", null);
                    }

                    Throwable cause = exception.getCause();
                    if (cause instanceof ResourceNotFoundException) {
                        throw new CompletionException("The workflow to delete was
not found.", cause);
```

```
                    }

                    // Wrap other AWS exceptions in a CompletionException.
                    throw new CompletionException("Failed to delete workflow: " +
exception.getMessage(), exception);
                }
            });
    }

    /**
     * Creates a schema mapping asynchronously.
     *
     * @param schemaName the name of the schema to create
     * @return a {@link CompletableFuture} that represents the asynchronous creation
of the schema mapping
     */
    public CompletableFuture<CreateSchemaMappingResponse>
createSchemaMappingAsync(String schemaName) {
        List<SchemaInputAttribute> schemaAttributes = null;
        if (schemaName.startsWith("json")) {
            schemaAttributes = List.of(

SchemaInputAttribute.builder().matchKey("id").fieldName("id").type(SchemaAttributeType.UNIQ

SchemaInputAttribute.builder().matchKey("name").fieldName("name").type(SchemaAttributeType.

SchemaInputAttribute.builder().matchKey("email").fieldName("email").type(SchemaAttributeTyp
            );
        } else {
            schemaAttributes = List.of(

SchemaInputAttribute.builder().matchKey("id").fieldName("id").type(SchemaAttributeType.UNIQ

SchemaInputAttribute.builder().matchKey("name").fieldName("name").type(SchemaAttributeType.

SchemaInputAttribute.builder().matchKey("email").fieldName("email").type(SchemaAttributeTyp

SchemaInputAttribute.builder().fieldName("phone").type(SchemaAttributeType.PROVIDER_ID).sub
            );
        }

        CreateSchemaMappingRequest request = CreateSchemaMappingRequest.builder()
            .schemaName(schemaName)
            .mappedInputFields(schemaAttributes)
```

```
            .build();

        return getResolutionAsyncClient().createSchemaMapping(request)
            .whenComplete((response, exception) -> {
                if (response != null) {
                    logger.info("[{}] schema mapping Created Successfully!",
schemaName);
                } else {
                    if (exception == null) {
                        throw new CompletionException("An unknown error occurred
while creating the schema mapping.", null);
                    }

                    Throwable cause = exception.getCause();
                    if (cause instanceof ConflictException) {
                        throw new CompletionException("A conflicting schema mapping
already exists. Resolve conflicts before proceeding.", cause);
                    }

                    // Wrap other AWS exceptions in a CompletionException.
                    throw new CompletionException("Failed to create schema mapping:
" + exception.getMessage(), exception);
                }
            });
    }

    /**
     * Retrieves the schema mapping asynchronously.
     *
     * @param schemaName the name of the schema to retrieve the mapping for
     * @return a {@link CompletableFuture} that completes with the {@link
GetSchemaMappingResponse} when the operation
     * is complete
     * @throws RuntimeException if the schema mapping retrieval fails
     */
    public CompletableFuture<GetSchemaMappingResponse> getSchemaMappingAsync(String
schemaName) {
        GetSchemaMappingRequest mappingRequest = GetSchemaMappingRequest.builder()
            .schemaName(schemaName)
            .build();

        return getResolutionAsyncClient().getSchemaMapping(mappingRequest)
            .whenComplete((response, exception) -> {
                if (response != null) {
```

```
                    response.mappedInputFields().forEach(attribute ->
                        logger.info("Attribute Name: " + attribute.fieldName() +
                            ", Attribute Type: " + attribute.type().toString()));
                } else {
                    if (exception == null) {
                        throw new CompletionException("An unknown error occurred
while getting schema mapping.", null);
                    }

                    Throwable cause = exception.getCause();
                    if (cause instanceof ResourceNotFoundException) {
                        throw new CompletionException("The requested schema mapping
was not found.", cause);
                    }

                    // Wrap other exceptions in a CompletionException with the
message.
                    throw new CompletionException("Failed to get schema mapping: " +
exception.getMessage(), exception);
                }
            });
    }

    /**
     * Asynchronously retrieves a matching job based on the provided job ID and
workflow name.
     *
     * @param jobId        the ID of the job to retrieve
     * @param workflowName the name of the workflow associated with the job
     * @return a {@link CompletableFuture} that completes when the job information
is available or an exception occurs
     */
    public CompletableFuture<GetMatchingJobResponse> getMatchingJobAsync(String
jobId, String workflowName) {
        GetMatchingJobRequest request = GetMatchingJobRequest.builder()
            .jobId(jobId)
            .workflowName(workflowName)
            .build();

        return getResolutionAsyncClient().getMatchingJob(request)
            .whenComplete((response, exception) -> {
                if (response != null) {
                    // Successfully fetched the matching job details, log the job
status.
```

```
                    logger.info("Job status: " + response.status());
                    logger.info("Job details: " + response.toString());
                } else {
                    if (exception == null) {
                        throw new CompletionException("An unknown error occurred
while fetching the matching job.", null);
                    }

                    Throwable cause = exception.getCause();
                    if (cause instanceof ResourceNotFoundException) {
                        throw new CompletionException("The requested job could not
be found.", cause);
                    }

                    // Wrap other exceptions in a CompletionException with the
message.
                    throw new CompletionException("Error fetching matching job: " +
exception.getMessage(), exception);
                }
            });
    }


    /**
     * Starts a matching job asynchronously for the specified workflow name.
     *
     * @param workflowName the name of the workflow for which to start the matching
job
     * @return a {@link CompletableFuture} that completes with the job ID of the
started matching job, or an empty
     * string if the operation fails
     */
    public CompletableFuture<String> startMatchingJobAsync(String workflowName) {
        StartMatchingJobRequest jobRequest = StartMatchingJobRequest.builder()
            .workflowName(workflowName)
            .build();

        return getResolutionAsyncClient().startMatchingJob(jobRequest)
            .whenComplete((response, exception) -> {
                if (response != null) {
                    String jobId = response.jobId();
                    logger.info("Job ID: " + jobId);
                } else {
                    if (exception == null) {
```

```
                        throw new CompletionException("An unknown error occurred
while starting the job.", null);
                    }

                    Throwable cause = exception.getCause();
                    if (cause instanceof ConflictException) {
                        throw new CompletionException("The job is already running.
Resolve conflicts before starting a new job.", cause);
                    }

                    // Wrap other AWS exceptions in a CompletionException.
                    throw new CompletionException("Failed to start the job: " +
exception.getMessage(), exception);
                }
            })
            .thenApply(response -> response != null ? response.jobId() : "");
    }

    /**
     * Checks the status of a workflow asynchronously.
     *
     * @param jobId        the ID of the job to check
     * @param workflowName the name of the workflow to check
     * @return a CompletableFuture that resolves to a boolean value indicating
whether the workflow has completed
     * successfully
     */
    public CompletableFuture<GetMatchingJobResponse>
checkWorkflowStatusCompleteAsync(String jobId, String workflowName) {
        GetMatchingJobRequest request = GetMatchingJobRequest.builder()
            .jobId(jobId)
            .workflowName(workflowName)
            .build();

        return getResolutionAsyncClient().getMatchingJob(request)
            .whenComplete((response, exception) -> {
                if (response != null) {
                    // Process the response and log the job status.
                    logger.info("Job status: " + response.status());
                } else {
                    // Ensure exception is not null before accessing its cause.
                    if (exception == null) {
                        throw new CompletionException("An unknown error occurred
while checking job status.", null);
```

```
                    }

                    Throwable cause = exception.getCause();
                    if (cause instanceof ResourceNotFoundException) {
                        throw new CompletionException("The requested resource was
not found while checking the job status.", cause);
                    }

                    // Wrap other AWS exceptions in a CompletionException.
                    throw new CompletionException("Failed to check job status: " +
exception.getMessage(), exception);
                }
            });
    }

    /**
     * Creates an asynchronous CompletableFuture to manage the creation of a
matching workflow.
     *
     * @param roleARN                the AWS IAM role ARN to be used for the
workflow execution
     * @param workflowName           the name of the workflow to be created
     * @param outputBucket           the S3 bucket path where the workflow output
will be stored
     * @param jsonGlueTableArn       the ARN of the Glue Data Catalog table to be
used as the input source
     * @param jsonErSchemaMappingName the name of the schema to be used for the
input source
     * @return a CompletableFuture that, when completed, will return the ARN of the
created workflow
     */
    public CompletableFuture<String> createMatchingWorkflowAsync(
        String roleARN
        , String workflowName
        , String outputBucket
        , String jsonGlueTableArn
        , String jsonErSchemaMappingName
        , String csvGlueTableArn
        , String csvErSchemaMappingName) {

        InputSource jsonInputSource = InputSource.builder()
            .inputSourceARN(jsonGlueTableArn)
            .schemaName(jsonErSchemaMappingName)
            .applyNormalization(false)
```

```
            .build();

        InputSource csvInputSource = InputSource.builder()
            .inputSourceARN(csvGlueTableArn)
            .schemaName(csvErSchemaMappingName)
            .applyNormalization(false)
            .build();

        OutputAttribute idOutputAttribute = OutputAttribute.builder()
            .name("id")
            .build();

        OutputAttribute nameOutputAttribute = OutputAttribute.builder()
            .name("name")
            .build();

        OutputAttribute emailOutputAttribute = OutputAttribute.builder()
            .name("email")
            .build();

        OutputAttribute phoneOutputAttribute = OutputAttribute.builder()
            .name("phone")
            .build();

        OutputSource outputSource = OutputSource.builder()
            .outputS3Path("s3://" + outputBucket + "/eroutput")
            .output(idOutputAttribute, nameOutputAttribute, emailOutputAttribute,
phoneOutputAttribute)
            .applyNormalization(false)
            .build();

        ResolutionTechniques resolutionType = ResolutionTechniques.builder()
            .resolutionType(ResolutionType.ML_MATCHING)
            .build();

        CreateMatchingWorkflowRequest workflowRequest =
CreateMatchingWorkflowRequest.builder()
            .roleArn(roleARN)
            .description("Created by using the AWS SDK for Java")
            .workflowName(workflowName)
            .inputSourceConfig(List.of(jsonInputSource, csvInputSource))
            .outputSourceConfig(List.of(outputSource))
            .resolutionTechniques(resolutionType)
            .build();
```

```
        return getResolutionAsyncClient().createMatchingWorkflow(workflowRequest)
            .whenComplete((response, exception) -> {
                if (response != null) {
                    logger.info("Workflow created successfully.");
                } else {
                    Throwable cause = exception.getCause();
                    if (cause instanceof ValidationException) {
                        throw new CompletionException("Invalid request: Please check
input parameters.", cause);
                    }

                    if (cause instanceof ConflictException) {
                        throw new CompletionException("A conflicting workflow
already exists. Resolve conflicts before proceeding.", cause);
                    }
                    throw new CompletionException("Failed to create workflow: " +
exception.getMessage(), exception);
                }
            })
            .thenApply(CreateMatchingWorkflowResponse::workflowArn);
    }

    /**
     * Tags the specified schema mapping ARN.
     *
     * @param schemaMappingARN the ARN of the schema mapping to tag
     */
    public CompletableFuture<TagResourceResponse> tagEntityResource(String
schemaMappingARN) {
        Map<String, String> tags = new HashMap<>();
        tags.put("tag1", "tag1Value");
        tags.put("tag2", "tag2Value");

        TagResourceRequest request = TagResourceRequest.builder()
            .resourceArn(schemaMappingARN)
            .tags(tags)
            .build();

        return getResolutionAsyncClient().tagResource(request)
            .whenComplete((response, exception) -> {
                if (response != null) {
                    // Successfully tagged the resource, log the success message.
                    logger.info("Successfully tagged the resource.");
```

```
                } else {
                    if (exception == null) {
                        throw new CompletionException("An unknown error occurred
while tagging the resource.", null);
                    }

                    Throwable cause = exception.getCause();
                    if (cause instanceof ResourceNotFoundException) {
                        throw new CompletionException("The resource to tag was not
found.", cause);
                    }
                    throw new CompletionException("Failed to tag the resource: " +
exception.getMessage(), exception);
                }
            });
    }

    public CompletableFuture<JobMetrics> getJobInfo(String workflowName, String
jobId) {
        return getResolutionAsyncClient().getMatchingJob(b -> b
                .workflowName(workflowName)
                .jobId(jobId))
            .whenComplete((response, exception) -> {
                if (response != null) {
                    logger.info("Job metrics fetched successfully for jobId: " +
jobId);
                } else {
                    Throwable cause = exception.getCause();
                    if (cause instanceof ResourceNotFoundException) {
                        throw new CompletionException("Invalid request: Job id was
not found.", cause);
                    }
                    throw new CompletionException("Failed to fetch job info: " +
exception.getMessage(), exception);
                }
            })
            .thenApply(response -> response.metrics()); // Extract job metrics
    }

    /**
     * Uploads data to an Amazon S3 bucket asynchronously.
     *
     * @param bucketName the name of the S3 bucket to upload the data to
     * @param jsonData   the JSON data to be uploaded
```

```
     * @param csvData    the CSV data to be uploaded
     * @return a {@link CompletableFuture} representing both asynchronous operation
of uploading the data
     * @throws RuntimeException if an error occurs during the file upload
     */

   public void uploadInputData(String bucketName, String jsonData, String csvData)
{
       // Upload JSON data.
       String jsonKey = "jsonData/data.json";
       PutObjectRequest jsonUploadRequest = PutObjectRequest.builder()
           .bucket(bucketName)
           .key(jsonKey)
           .contentType("application/json")
           .build();

       CompletableFuture<PutObjectResponse> jsonUploadResponse =
getS3AsyncClient().putObject(jsonUploadRequest,
AsyncRequestBody.fromString(jsonData));

       // Upload CSV data.
       String csvKey = "csvData/data.csv";
       PutObjectRequest csvUploadRequest = PutObjectRequest.builder()
           .bucket(bucketName)
           .key(csvKey)
           .contentType("text/csv")
           .build();
       CompletableFuture<PutObjectResponse> csvUploadResponse =
getS3AsyncClient().putObject(csvUploadRequest,
AsyncRequestBody.fromString(csvData));

       CompletableFuture.allOf(jsonUploadResponse, csvUploadResponse)
           .whenComplete((result, ex) -> {
               if (ex != null) {
                   // Wrap an AWS exception.
                   throw new CompletionException("Failed to upload files", ex);
               }
           }).join();

   }

   /**
    * Finds the latest file in the S3 bucket that starts with "run-" in any depth
of subfolders
```

```
     */
    private CompletableFuture<String> findLatestMatchingFile(String bucketName) {
        ListObjectsV2Request request = ListObjectsV2Request.builder()
            .bucket(bucketName)
            .prefix(PREFIX) // Searches within the given folder
            .build();

        return getS3AsyncClient().listObjectsV2(request)
            .thenApply(response -> response.contents().stream()
                .map(S3Object::key)
                .filter(key -> key.matches(".*?/run-[0-9a-zA-Z\\-]+")) // Matches
files like run-XXXXX in any subfolder
                .max(String::compareTo) // Gets the latest file
                .orElse(null))
            .whenComplete((result, exception) -> {
                if (exception == null) {
                    if (result != null) {
                        logger.info("Latest matching file found: " + result);
                    } else {
                        logger.info("No matching files found.");
                    }
                } else {
                    throw new CompletionException("Failed to find latest matching
file: " + exception.getMessage(), exception);
                }
            });
    }

    /**
     * Prints the data located in the file in the S3 bucket that starts with "run-"
in any depth of subfolders
     */
    public void printData(String bucketName) {
        try {
            // Find the latest file with "run-" prefix in any depth of subfolders.
            String s3Key = findLatestMatchingFile(bucketName).join();
            if (s3Key == null) {
                logger.error("No matching files found in S3.");
                return;
            }

            logger.info("Downloading file: " + s3Key);

            // Read CSV file as String.
```

```
            String csvContent = readCSVFromS3Async(bucketName, s3Key).join();
            if (csvContent.isEmpty()) {
                logger.error("File is empty.");
                return;
            }

            // Process CSV content.
            List<String[]> records = parseCSV(csvContent);
            printTable(records);

        } catch (RuntimeException | IOException | CsvException e) {
            logger.error("Error processing CSV file from S3: " + e.getMessage());
            e.printStackTrace();
        }
    }

    /**
     * Reads a CSV file from S3 and returns it as a String.
     */
    private static CompletableFuture<String> readCSVFromS3Async(String bucketName,
String s3Key) {
        GetObjectRequest getObjectRequest = GetObjectRequest.builder()
            .bucket(bucketName)
            .key(s3Key)
            .build();

        // Initiating the asynchronous request to get the file as bytes
        return getS3AsyncClient().getObject(getObjectRequest,
AsyncResponseTransformer.toBytes())
            .thenApply(responseBytes -> responseBytes.asUtf8String()) // Convert
bytes to UTF-8 string
            .whenComplete((result, exception) -> {
                if (exception != null) {
                    throw new CompletionException("Failed to read CSV from S3: " +
exception.getMessage(), exception);
                } else {
                    logger.info("Successfully fetched CSV file content from S3.");
                }
            });
    }

    /**
     * Parses CSV content from a String into a list of records.
     */
```

```
   private static List<String[]> parseCSV(String csvContent) throws IOException,
CsvException {
       try (CSVReader csvReader = new CSVReader(new StringReader(csvContent))) {
           return csvReader.readAll();
       }
   }

   /**
    * Prints the given CSV data in a formatted table
    */
   private static void printTable(List<String[]> records) {
       if (records.isEmpty()) {
           System.out.println("No records found.");
           return;
       }

       String[] headers = records.get(0);
       List<String[]> rows = records.subList(1, records.size());

       // Determine column widths dynamically based on longest content
       int[] columnWidths = new int[headers.length];
       for (int i = 0; i < headers.length; i++) {
           final int columnIndex = i;
           int maxWidth = Math.max(headers[i].length(), rows.stream()
               .map(row -> row.length > columnIndex ? row[columnIndex].length() :
0)
               .max(Integer::compareTo)
               .orElse(0));
           columnWidths[i] = Math.min(maxWidth, 25); // Limit max width for better
readability
       }

       // Enable ANSI Console for colored output
       AnsiConsole.systemInstall();

       // Print table header
       System.out.println(ansi().fgYellow().a("=== CSV Data from S3 ===").reset());
       printRow(headers, columnWidths, true);

       // Print rows
       rows.forEach(row -> printRow(row, columnWidths, false));

       // Restore console to normal
       AnsiConsole.systemUninstall();
```

```
    }

    private static void printRow(String[] row, int[] columnWidths, boolean isHeader)
 {
        String border = IntStream.range(0, columnWidths.length)
            .mapToObj(i -> "-".repeat(columnWidths[i] + 2))
            .collect(Collectors.joining("+", "+", "+"));

        if (isHeader) {
            System.out.println(border);
        }

        System.out.print("|");
        for (int i = 0; i < columnWidths.length; i++) {
            String cell = (i < row.length && row[i] != null) ? row[i] : "";
            System.out.printf(" %-" + columnWidths[i] + "s |", isHeader ?
 ansi().fgBrightBlue().a(cell).reset() : cell);
        }
        System.out.println();

        if (isHeader) {
            System.out.println(border);
        }
    }
}
```

Azioni

`CheckWorkflowStatus`

Il seguente esempio di codice mostra come usare. `CheckWorkflowStatus`

SDK per Java 2.x

> **Note**
>
> C'è 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 status of a workflow asynchronously.
    *
    * @param jobId        the ID of the job to check
    * @param workflowName the name of the workflow to check
    * @return a CompletableFuture that resolves to a boolean value indicating
whether the workflow has completed
    * successfully
    */
   public CompletableFuture<GetMatchingJobResponse>
checkWorkflowStatusCompleteAsync(String jobId, String workflowName) {
       GetMatchingJobRequest request = GetMatchingJobRequest.builder()
           .jobId(jobId)
           .workflowName(workflowName)
           .build();

       return getResolutionAsyncClient().getMatchingJob(request)
           .whenComplete((response, exception) -> {
               if (response != null) {
                   // Process the response and log the job status.
                   logger.info("Job status: " + response.status());
               } else {
                   // Ensure exception is not null before accessing its cause.
                   if (exception == null) {
                       throw new CompletionException("An unknown error occurred
while checking job status.", null);
                   }

                   Throwable cause = exception.getCause();
                   if (cause instanceof ResourceNotFoundException) {
                       throw new CompletionException("The requested resource was
not found while checking the job status.", cause);
                   }

                   // Wrap other AWS exceptions in a CompletionException.
                   throw new CompletionException("Failed to check job status: " +
exception.getMessage(), exception);
               }
           });
   }
```

- Per i dettagli sull'API, CheckWorkflowStatusconsulta AWS SDK for Java 2.x API Reference.

`CreateMatchingWorkflow`

Il seguente esempio di codice mostra come utilizzare`CreateMatchingWorkflow`.

SDK per Java 2.x

> **Note**
>
> C'è 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 asynchronous CompletableFuture to manage the creation of a
matching workflow.
    *
    * @param roleARN                the AWS IAM role ARN to be used for the
workflow execution
    * @param workflowName           the name of the workflow to be created
    * @param outputBucket           the S3 bucket path where the workflow output
will be stored
    * @param jsonGlueTableArn       the ARN of the Glue Data Catalog table to be
used as the input source
    * @param jsonErSchemaMappingName the name of the schema to be used for the
input source
    * @return a CompletableFuture that, when completed, will return the ARN of the
created workflow
    */
   public CompletableFuture<String> createMatchingWorkflowAsync(
       String roleARN
       , String workflowName
       , String outputBucket
       , String jsonGlueTableArn
       , String jsonErSchemaMappingName
       , String csvGlueTableArn
       , String csvErSchemaMappingName) {

       InputSource jsonInputSource = InputSource.builder()
           .inputSourceARN(jsonGlueTableArn)
           .schemaName(jsonErSchemaMappingName)
           .applyNormalization(false)
           .build();
```

```
        InputSource csvInputSource = InputSource.builder()
            .inputSourceARN(csvGlueTableArn)
            .schemaName(csvErSchemaMappingName)
            .applyNormalization(false)
            .build();

        OutputAttribute idOutputAttribute = OutputAttribute.builder()
            .name("id")
            .build();

        OutputAttribute nameOutputAttribute = OutputAttribute.builder()
            .name("name")
            .build();

        OutputAttribute emailOutputAttribute = OutputAttribute.builder()
            .name("email")
            .build();

        OutputAttribute phoneOutputAttribute = OutputAttribute.builder()
            .name("phone")
            .build();

        OutputSource outputSource = OutputSource.builder()
            .outputS3Path("s3://" + outputBucket + "/eroutput")
            .output(idOutputAttribute, nameOutputAttribute, emailOutputAttribute,
phoneOutputAttribute)
            .applyNormalization(false)
            .build();

        ResolutionTechniques resolutionType = ResolutionTechniques.builder()
            .resolutionType(ResolutionType.ML_MATCHING)
            .build();

        CreateMatchingWorkflowRequest workflowRequest =
CreateMatchingWorkflowRequest.builder()
            .roleArn(roleARN)
            .description("Created by using the AWS SDK for Java")
            .workflowName(workflowName)
            .inputSourceConfig(List.of(jsonInputSource, csvInputSource))
            .outputSourceConfig(List.of(outputSource))
            .resolutionTechniques(resolutionType)
            .build();

        return getResolutionAsyncClient().createMatchingWorkflow(workflowRequest)
```

```
            .whenComplete((response, exception) -> {
                if (response != null) {
                    logger.info("Workflow created successfully.");
                } else {
                    Throwable cause = exception.getCause();
                    if (cause instanceof ValidationException) {
                        throw new CompletionException("Invalid request: Please check
input parameters.", cause);
                    }

                    if (cause instanceof ConflictException) {
                        throw new CompletionException("A conflicting workflow
already exists. Resolve conflicts before proceeding.", cause);
                    }
                    throw new CompletionException("Failed to create workflow: " +
exception.getMessage(), exception);
                }
            })
            .thenApply(CreateMatchingWorkflowResponse::workflowArn);
    }
```

- Per i dettagli sull'API, CreateMatchingWorkflowconsulta AWS SDK for Java 2.x API Reference.

`CreateSchemaMapping`

Il seguente esempio di codice mostra come utilizzare`CreateSchemaMapping`.

SDK per Java 2.x

> **Note**
>
> C'è 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 a schema mapping asynchronously.
     *
     * @param schemaName the name of the schema to create
     * @return a {@link CompletableFuture} that represents the asynchronous creation
of the schema mapping
```

```
     */
    public CompletableFuture<CreateSchemaMappingResponse>
createSchemaMappingAsync(String schemaName) {
        List<SchemaInputAttribute> schemaAttributes = null;
        if (schemaName.startsWith("json")) {
            schemaAttributes = List.of(

SchemaInputAttribute.builder().matchKey("id").fieldName("id").type(SchemaAttributeType.UNIQ

SchemaInputAttribute.builder().matchKey("name").fieldName("name").type(SchemaAttributeType.

SchemaInputAttribute.builder().matchKey("email").fieldName("email").type(SchemaAttributeTyp
            );
        } else {
            schemaAttributes = List.of(

SchemaInputAttribute.builder().matchKey("id").fieldName("id").type(SchemaAttributeType.UNIQ

SchemaInputAttribute.builder().matchKey("name").fieldName("name").type(SchemaAttributeType.

SchemaInputAttribute.builder().matchKey("email").fieldName("email").type(SchemaAttributeTyp

SchemaInputAttribute.builder().fieldName("phone").type(SchemaAttributeType.PROVIDER_ID).sub
            );
        }

        CreateSchemaMappingRequest request = CreateSchemaMappingRequest.builder()
            .schemaName(schemaName)
            .mappedInputFields(schemaAttributes)
            .build();

        return getResolutionAsyncClient().createSchemaMapping(request)
            .whenComplete((response, exception) -> {
                if (response != null) {
                    logger.info("[{}] schema mapping Created Successfully!",
schemaName);
                } else {
                    if (exception == null) {
                        throw new CompletionException("An unknown error occurred
while creating the schema mapping.", null);
                    }

                    Throwable cause = exception.getCause();
                    if (cause instanceof ConflictException) {
```

```
                        throw new CompletionException("A conflicting schema mapping
already exists. Resolve conflicts before proceeding.", cause);
                    }

                    // Wrap other AWS exceptions in a CompletionException.
                    throw new CompletionException("Failed to create schema mapping:
" + exception.getMessage(), exception);
                }
            });
    }
```

- Per i dettagli sull'API, CreateSchemaMappingconsulta AWS SDK for Java 2.x API Reference.

`DeleteMatchingWorkflow`

Il seguente esempio di codice mostra come utilizzare`DeleteMatchingWorkflow`.

SDK per Java 2.x

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
    /**
     * Asynchronously deletes a workflow with the specified name.
     *
     * @param workflowName the name of the workflow to be deleted
     * @return a {@link CompletableFuture} that completes when the workflow has been
deleted
     * @throws RuntimeException if the deletion of the workflow fails
     */
    public CompletableFuture<DeleteMatchingWorkflowResponse>
deleteMatchingWorkflowAsync(String workflowName) {
        DeleteMatchingWorkflowRequest request =
DeleteMatchingWorkflowRequest.builder()
            .workflowName(workflowName)
            .build();
```

```
        return getResolutionAsyncClient().deleteMatchingWorkflow(request)
            .whenComplete((response, exception) -> {
                if (response != null) {
                    logger.info("{} was deleted", workflowName );
                } else {
                    if (exception == null) {
                        throw new CompletionException("An unknown error occurred
while deleting the workflow.", null);
                    }

                    Throwable cause = exception.getCause();
                    if (cause instanceof ResourceNotFoundException) {
                        throw new CompletionException("The workflow to delete was
not found.", cause);
                    }

                    // Wrap other AWS exceptions in a CompletionException.
                    throw new CompletionException("Failed to delete workflow: " +
exception.getMessage(), exception);
                }
            });
    }
```

- Per i dettagli sull'API, DeleteMatchingWorkflowconsulta AWS SDK for Java 2.x API Reference.

`DeleteSchemaMapping`

Il seguente esempio di codice mostra come utilizzare`DeleteSchemaMapping`.

SDK per Java 2.x

> **Note**
>
> C'è 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 schema mapping asynchronously.
     *
     * @param schemaName the name of the schema to delete
```

```
    * @return a {@link CompletableFuture} that completes when the schema mapping is
deleted successfully,
    * or throws a {@link RuntimeException} if the deletion fails
    */
   public CompletableFuture<DeleteSchemaMappingResponse>
deleteSchemaMappingAsync(String schemaName) {
       DeleteSchemaMappingRequest request = DeleteSchemaMappingRequest.builder()
           .schemaName(schemaName)
           .build();

       return getResolutionAsyncClient().deleteSchemaMapping(request)
           .whenComplete((response, exception) -> {
               if (response != null) {
                   // Successfully deleted the schema mapping, log the success
message.
                   logger.info("Schema mapping '{}' deleted successfully.",
schemaName);
               } else {
                   // Ensure exception is not null before accessing its cause.
                   if (exception == null) {
                       throw new CompletionException("An unknown error occurred
while deleting the schema mapping.", null);
                   }

                   Throwable cause = exception.getCause();
                   if (cause instanceof ResourceNotFoundException) {
                       throw new CompletionException("The schema mapping was not
found to delete: " + schemaName, cause);
                   }

                   // Wrap other AWS exceptions in a CompletionException.
                   throw new CompletionException("Failed to delete schema mapping:
" + schemaName, exception);
               }
           });
   }
```

- Per i dettagli sull'API, DeleteSchemaMappingconsulta AWS SDK for Java 2.x API Reference.

`GetMatchingJob`

Il seguente esempio di codice mostra come utilizzare`GetMatchingJob`.

SDK per Java 2.x

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
   /**
    * Asynchronously retrieves a matching job based on the provided job ID and
workflow name.
    *
    * @param jobId        the ID of the job to retrieve
    * @param workflowName the name of the workflow associated with the job
    * @return a {@link CompletableFuture} that completes when the job information
is available or an exception occurs
    */
   public CompletableFuture<GetMatchingJobResponse> getMatchingJobAsync(String
jobId, String workflowName) {
       GetMatchingJobRequest request = GetMatchingJobRequest.builder()
           .jobId(jobId)
           .workflowName(workflowName)
           .build();

       return getResolutionAsyncClient().getMatchingJob(request)
           .whenComplete((response, exception) -> {
               if (response != null) {
                   // Successfully fetched the matching job details, log the job
status.
                   logger.info("Job status: " + response.status());
                   logger.info("Job details: " + response.toString());
               } else {
                   if (exception == null) {
                       throw new CompletionException("An unknown error occurred
while fetching the matching job.", null);
                   }

                   Throwable cause = exception.getCause();
                   if (cause instanceof ResourceNotFoundException) {
                       throw new CompletionException("The requested job could not
be found.", cause);
                   }
```

```
                    // Wrap other exceptions in a CompletionException with the
message.
                    throw new CompletionException("Error fetching matching job: " +
exception.getMessage(), exception);
                }
            });
    }
```

- Per i dettagli sull'API, GetMatchingJobconsulta AWS SDK for Java 2.x API Reference.

`GetSchemaMapping`

Il seguente esempio di codice mostra come utilizzare`GetSchemaMapping`.

SDK per Java 2.x

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
    /**
     * Retrieves the schema mapping asynchronously.
     *
     * @param schemaName the name of the schema to retrieve the mapping for
     * @return a {@link CompletableFuture} that completes with the {@link
GetSchemaMappingResponse} when the operation
     * is complete
     * @throws RuntimeException if the schema mapping retrieval fails
     */
    public CompletableFuture<GetSchemaMappingResponse> getSchemaMappingAsync(String
schemaName) {
        GetSchemaMappingRequest mappingRequest = GetSchemaMappingRequest.builder()
            .schemaName(schemaName)
            .build();

        return getResolutionAsyncClient().getSchemaMapping(mappingRequest)
            .whenComplete((response, exception) -> {
```

```
                if (response != null) {
                    response.mappedInputFields().forEach(attribute ->
                        logger.info("Attribute Name: " + attribute.fieldName() +
                            ", Attribute Type: " + attribute.type().toString()));
                } else {
                    if (exception == null) {
                        throw new CompletionException("An unknown error occurred
while getting schema mapping.", null);
                    }

                    Throwable cause = exception.getCause();
                    if (cause instanceof ResourceNotFoundException) {
                        throw new CompletionException("The requested schema mapping
was not found.", cause);
                    }

                    // Wrap other exceptions in a CompletionException with the
message.
                    throw new CompletionException("Failed to get schema mapping: " +
exception.getMessage(), exception);
                }
            });
    }
```

- Per i dettagli sull'API, GetSchemaMappingconsulta AWS SDK for Java 2.x API Reference.

`ListSchemaMappings`

Il seguente esempio di codice mostra come utilizzare`ListSchemaMappings`.

SDK per Java 2.x

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
    /**
     * Lists the schema mappings associated with the current AWS account. This
method uses an asynchronous paginator to
```

```
    * retrieve the schema mappings, and prints the name of each schema mapping to
the console.
    */
   public void ListSchemaMappings() {
       ListSchemaMappingsRequest mappingsRequest =
ListSchemaMappingsRequest.builder()
           .build();

       ListSchemaMappingsPublisher paginator =
getResolutionAsyncClient().listSchemaMappingsPaginator(mappingsRequest);

       // Iterate through the pages of results
       CompletableFuture<Void> future = paginator.subscribe(response -> {
           response.schemaList().forEach(schemaMapping ->
               logger.info("Schema Mapping Name: " + schemaMapping.schemaName())
           );
       });

       // Wait for the asynchronous operation to complete
       future.join();
   }
```

- Per i dettagli sull'API, ListSchemaMappingsconsulta AWS SDK for Java 2.x API Reference.

`StartMatchingJob`

Il seguente esempio di codice mostra come utilizzare`StartMatchingJob`.

SDK per Java 2.x

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
   /**
    * Starts a matching job asynchronously for the specified workflow name.
    *
```

```
    * @param workflowName the name of the workflow for which to start the matching
job
    * @return a {@link CompletableFuture} that completes with the job ID of the
started matching job, or an empty
    * string if the operation fails
    */
   public CompletableFuture<String> startMatchingJobAsync(String workflowName) {
       StartMatchingJobRequest jobRequest = StartMatchingJobRequest.builder()
           .workflowName(workflowName)
           .build();

       return getResolutionAsyncClient().startMatchingJob(jobRequest)
           .whenComplete((response, exception) -> {
               if (response != null) {
                   String jobId = response.jobId();
                   logger.info("Job ID: " + jobId);
               } else {
                   if (exception == null) {
                       throw new CompletionException("An unknown error occurred
while starting the job.", null);
                   }

                   Throwable cause = exception.getCause();
                   if (cause instanceof ConflictException) {
                       throw new CompletionException("The job is already running.
Resolve conflicts before starting a new job.", cause);
                   }

                   // Wrap other AWS exceptions in a CompletionException.
                   throw new CompletionException("Failed to start the job: " +
exception.getMessage(), exception);
               }
           })
           .thenApply(response -> response != null ? response.jobId() : "");
   }
```

- Per i dettagli sull'API, StartMatchingJobconsulta AWS SDK for Java 2.x API Reference.

`TagEntityResource`

Il seguente esempio di codice mostra come utilizzare`TagEntityResource`.

SDK per Java 2.x

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
   /**
    * Tags the specified schema mapping ARN.
    *
    * @param schemaMappingARN the ARN of the schema mapping to tag
    */
   public CompletableFuture<TagResourceResponse> tagEntityResource(String
schemaMappingARN) {
       Map<String, String> tags = new HashMap<>();
       tags.put("tag1", "tag1Value");
       tags.put("tag2", "tag2Value");

       TagResourceRequest request = TagResourceRequest.builder()
           .resourceArn(schemaMappingARN)
           .tags(tags)
           .build();

       return getResolutionAsyncClient().tagResource(request)
           .whenComplete((response, exception) -> {
               if (response != null) {
                   // Successfully tagged the resource, log the success message.
                   logger.info("Successfully tagged the resource.");
               } else {
                   if (exception == null) {
                       throw new CompletionException("An unknown error occurred
while tagging the resource.", null);
                   }

                   Throwable cause = exception.getCause();
                   if (cause instanceof ResourceNotFoundException) {
                       throw new CompletionException("The resource to tag was not
found.", cause);
                   }
                   throw new CompletionException("Failed to tag the resource: " +
exception.getMessage(), exception);
```

```
                }
            });
    }
```

- Per i dettagli sull'API, TagEntityResourceconsulta 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 nozioni di base sono esempi di codice che mostrano come eseguire le operazioni essenziali all'interno di un servizio.

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.

Ogni esempio include un collegamento al codice sorgente completo, in cui è possibile trovare istruzioni su come configurare ed eseguire il codice nel contesto.

Nozioni di base

Hello OpenSearch Service

Il seguente esempio di codice mostra come iniziare a utilizzare OpenSearch Service.

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.services.opensearch.OpenSearchAsyncClient;
import software.amazon.awssdk.services.opensearch.model.ListVersionsRequest;
import java.util.List;
import java.util.concurrent.CompletableFuture;
```

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class HelloOpenSearch {
    public static void main(String[] args) {
        try {
            CompletableFuture<Void> future = listVersionsAsync();
            future.join();
            System.out.println("Versions listed successfully.");
        } catch (RuntimeException e) {
            System.err.println("Error occurred while listing versions: " +
 e.getMessage());
        }
    }

    private static OpenSearchAsyncClient getAsyncClient() {
        return OpenSearchAsyncClient.builder().build();
    }

    public static CompletableFuture<Void> listVersionsAsync() {
        ListVersionsRequest request = ListVersionsRequest.builder()
            .maxResults(10)
            .build();

        return getAsyncClient().listVersions(request).thenAccept(response -> {
            List<String> versionList = response.versions();
            for (String version : versionList) {
                System.out.println("Version info: " + version);
            }
        }).exceptionally(ex -> {
            // Handle the exception, or propagate it as a RuntimeException
            throw new RuntimeException("Failed to list versions", ex);
        });
    }
}
```

- Per i dettagli sull'API, ListVersionsconsulta AWS SDK for Java 2.x API Reference.

Argomenti

- Nozioni di base
- Azioni

Nozioni di base

Scopri le operazioni principali del OpenSearch servizio

L'esempio di codice seguente mostra come:

- Crea un dominio OpenSearch di servizio.
- Fornisce informazioni dettagliate su un dominio OpenSearch di servizio specifico.
- Elenca tutti i domini di OpenSearch servizio di proprietà dell'account.
- Attende che lo stato di modifica del dominio di OpenSearch servizio raggiunga lo stato completo.
- Modifica la configurazione di un dominio di OpenSearch servizio esistente.
- Aggiunge un tag al dominio del OpenSearch servizio.
- Elenca i tag associati a un dominio OpenSearch di servizio.
- Rimuove i tag da un dominio OpenSearch di servizio.
- Elimina il dominio del OpenSearch servizio.

SDK per Java 2.x

> **Note**
>
> C'è altro da fare. 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 OpenSearch del servizio.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.opensearch.model.*;
import java.util.List;
import java.util.Scanner;
import java.util.concurrent.CompletableFuture;
```

```
public class OpenSearchScenario {

    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    private static final Logger logger =
 LoggerFactory.getLogger(OpenSearchScenario.class);
    static Scanner scanner = new Scanner(System.in);

    static OpenSearchActions openSearchActions = new OpenSearchActions();

    public static void main(String[] args) throws Throwable {
        logger.info("""
            Welcome to the Amazon OpenSearch Service Basics Scenario.

            Use the Amazon OpenSearch Service API to create, configure, and manage
 OpenSearch Service domains.

            The operations exposed by the AWS OpenSearch Service client are focused
 on managing the OpenSearch Service domains
            and their configurations, not the data within the domains (such as
 indexing or querying documents).
            For document management, you typically interact directly with the
 OpenSearch REST API or use other libraries,
            such as the OpenSearch Java client (https://opensearch.org/docs/latest/
clients/java/).

            Let's get started...
        """);
        waitForInputToContinue(scanner);
        try {
            runScenario();
        } catch (RuntimeException e) {
            e.printStackTrace();
        }
    }

    private static void waitForInputToContinue(Scanner scanner) {
        while (true) {
            logger.info("");
            logger.info("Enter 'c' followed by <ENTER> to continue:");
            String input = scanner.nextLine();

            if (input.trim().equalsIgnoreCase("c")) {
                logger.info("Continuing with the program...");
```

```
                logger.info("");
                break;
            } else {
                logger.info("Invalid input. Please try again.");
            }
        }
    }

    private static void runScenario() throws Throwable {
        String currentTimestamp = String.valueOf(System.currentTimeMillis());
        String domainName = "test-domain-" + currentTimestamp;

        logger.info(DASHES);
        logger.info("1. Create an Amazon OpenSearch domain");
        logger.info("""
            An Amazon OpenSearch domain is a managed instance of the OpenSearch
engine,
            which is an open-source search and analytics engine derived from
Elasticsearch.
            An OpenSearch domain is essentially a cluster of compute resources and
storage that hosts
            one or more OpenSearch indexes, enabling you to perform full-text
searches, data analysis, and
            visualizations.

            In this step, we'll initiate the creation of the domain. We'll check on
the progress in a later step.
        """);
        waitForInputToContinue(scanner);

        try {
            CompletableFuture<String> future =
openSearchActions.createNewDomainAsync(domainName);
            String domainId = future.join();
            logger.info("Domain successfully created with ID: {}", domainId);
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause != null) {
                if (cause instanceof OpenSearchException openSearchEx) {
                    logger.error("OpenSearch error occurred: Error message:
{}, Error code {}", openSearchEx.awsErrorDetails().errorMessage(),
openSearchEx.awsErrorDetails().errorCode());
                } else {
```

```
                    logger.error("An unexpected error occurred: " +
cause.getMessage(), cause);
                }
            } else {
                logger.error("An unexpected error occurred: " + rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info("2. Describe the Amazon OpenSearch domain");
        logger.info("In this step, we get back the Domain ARN which is used in an
upcoming step.");
        waitForInputToContinue(scanner);

        String arn = "";
        try {
            CompletableFuture<String> future =
openSearchActions.describeDomainAsync(domainName);
            arn = future.join();
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof OpenSearchException openSearchEx) {
                logger.info("OpenSearch error occurred: Error message:
{}, Error code {}", openSearchEx.awsErrorDetails().errorMessage(),
openSearchEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info("3. List the domains in your account");
        waitForInputToContinue(scanner);

        try {
            CompletableFuture<List<DomainInfo>> future =
openSearchActions.listAllDomainsAsync();
            List<DomainInfo> domainInfoList = future.join();
            for (DomainInfo domain : domainInfoList) {
                logger.info("Domain name is: " + domain.domainName());
```

```
            }
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            while (cause.getCause() != null && !(cause instanceof
OpenSearchException)) {
                cause = cause.getCause();
            }
            if (cause instanceof OpenSearchException openSearchEx) {
                logger.info("OpenSearch error occurred: Error message:
{}, Error code {}", openSearchEx.awsErrorDetails().errorMessage(),
openSearchEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            throw cause;
        }

        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info("4. Wait until the domain's change status reaches a completed
state");
        logger.info("""
            In this step, we check on the change status of the domain that we
initiated in Step 1.
            Until we reach a COMPLETED state, we stay in a loop by sending a
DescribeDomainChangeProgressRequest.

            The time it takes for a change to an OpenSearch domain to reach a
completed state can range
            from a few minutes to several hours. In this case the change is creating
a new domain that we initiated in Step 1.
            The time varies depending on the complexity of the change and the
current load on
            the OpenSearch service. In general, simple changes, such as scaling the
number of data nodes or
            updating the OpenSearch version, may take 10-30 minutes.
        """);

        waitForInputToContinue(scanner);

        try {
            CompletableFuture<Void> future =
openSearchActions.domainChangeProgressAsync(domainName);
```

```
            future.join();
            logger.info("Domain change progress completed successfully.");
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            while (cause.getCause() != null && !(cause instanceof
ResourceNotFoundException)) {
                cause = cause.getCause();
            }
            if (cause instanceof ResourceNotFoundException
resourceNotFoundException) {
                logger.info("The specific AWS resource was not found: Error message:
{}, Error code {}", resourceNotFoundException.awsErrorDetails().errorMessage(),
resourceNotFoundException.awsErrorDetails().errorCode());

                if (cause instanceof OpenSearchException ex) {
                    logger.info("An OpenSearch error occurred: Error message: " +
ex.getMessage());
                } else {
                    logger.info("An unexpected error occurred: " + rt.getMessage());
                }
                throw cause;
            }
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info("5. Modify the domain");
        logger.info("""
            You can change your OpenSearch domain's settings, like the number of
instances, without starting over from scratch.
            This makes it easy to adjust your domain as your needs change, allowing
you to scale up or
            down quickly without recreating everything.

            We modify the domain in this step by changing the number of instances.
        """);

        waitForInputToContinue(scanner);

        try {
            CompletableFuture<UpdateDomainConfigResponse> future =
openSearchActions.updateSpecificDomainAsync(domainName);
            UpdateDomainConfigResponse updateResponse = future.join();
```

```
            logger.info("Domain update status: " +
updateResponse.domainConfig().changeProgressDetails().configChangeStatusAsString());
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof OpenSearchException openSearchEx) {
                logger.info("OpenSearch error occurred: Error message:
{}, Error code {}", openSearchEx.awsErrorDetails().errorMessage(),
openSearchEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info("6. Wait until the domain's change status reaches a completed
state");
        logger.info("""
            In this step, we poll the status until the domain's change status
reaches a completed state.
        """);

        waitForInputToContinue(scanner);

        try {
            CompletableFuture<Void> future =
openSearchActions.domainChangeProgressAsync(domainName);
            future.join();
            logger.info("Domain change progress completed successfully.");
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof OpenSearchException ex) {
                logger.info("EC2 error occurred: Error message: " +ex.getMessage());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info("7. Tag the Domain");
        logger.info("""
```

```
            Tags let you assign arbitrary information to an Amazon OpenSearch
Service domain so you can
            categorize and filter on that information. A tag is a key-value pair
that you define and
            associate with an OpenSearch Service domain. You can use these tags to
track costs by grouping
            expenses for similarly tagged resources.

            In this scenario, we create tags with keys "service" and "instances".
        """);

        waitForInputToContinue(scanner);

        try {
            CompletableFuture<AddTagsResponse> future =
openSearchActions.addDomainTagsAsync(arn);
            future.join();
            logger.info("Domain tags added successfully.");
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            while (cause.getCause() != null && !(cause instanceof
OpenSearchException)) {
                cause = cause.getCause();
            }
            if (cause instanceof OpenSearchException openSearchEx) {
                logger.info("OpenSearch error occurred: Error message:
{}, Error code {}", openSearchEx.awsErrorDetails().errorMessage(),
openSearchEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
                if (cause != null) {
                    if (cause instanceof OpenSearchException) {
                        logger.error("OpenSearch error occurred: Error message: " +
cause.getMessage(), cause);
                    } else {
                        logger.error("An unexpected error occurred: " +
cause.getMessage(), cause);
                    }
                } else {
                    logger.error("An unexpected error occurred: " + rt.getMessage(),
rt);
                }
                throw cause;
            }
```

```
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info("8. List Domain tags");
        waitForInputToContinue(scanner);

        try {
            CompletableFuture<ListTagsResponse> future =
openSearchActions.listDomainTagsAsync(arn);
            ListTagsResponse listTagsResponse = future.join();
            listTagsResponse.tagList().forEach(tag -> logger.info("Tag Key: " +
tag.key() + ", Tag Value: " + tag.value()));
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            while (cause.getCause() != null && !(cause instanceof
OpenSearchException)) {
                cause = cause.getCause();
            }
            if (cause instanceof OpenSearchException openSearchEx) {
                logger.info("OpenSearch error occurred: Error message:
{}, Error code {}", openSearchEx.awsErrorDetails().errorMessage(),
openSearchEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            throw cause;

        }

        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info("9. Delete the domain");
        logger.info("""
            In this step, we'll delete the Amazon OpenSearch domain that we created
in Step 1.
            Deleting a domain will remove all data and configuration for that
domain.
        """);

        waitForInputToContinue(scanner);

        try {
```

```
            CompletableFuture<DeleteDomainResponse> future =
openSearchActions.deleteSpecificDomainAsync(domainName);
            future.join();
            logger.info("Domain successfully deleted.");
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            while (cause.getCause() != null && !(cause instanceof
OpenSearchException)) {
                cause = cause.getCause();
            }
            if (cause instanceof OpenSearchException openSearchEx) {
                logger.info("OpenSearch error occurred: Error message:
{}, Error code {}", openSearchEx.awsErrorDetails().errorMessage(),
openSearchEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            throw cause;

        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info("Scenario complete!");
    }
}
```

Una classe wrapper per i metodi OpenSearch Service SDK.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.client.config.ClientOverrideConfiguration;
import software.amazon.awssdk.core.retry.RetryPolicy;
import software.amazon.awssdk.http.async.SdkAsyncHttpClient;
import software.amazon.awssdk.http.nio.netty.NettyNioAsyncHttpClient;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.opensearch.OpenSearchAsyncClient;
import software.amazon.awssdk.services.opensearch.model.AddTagsRequest;
import software.amazon.awssdk.services.opensearch.model.AddTagsResponse;
import software.amazon.awssdk.services.opensearch.model.ClusterConfig;
import software.amazon.awssdk.services.opensearch.model.CreateDomainRequest;
import software.amazon.awssdk.services.opensearch.model.DeleteDomainRequest;
```

```
import software.amazon.awssdk.services.opensearch.model.DeleteDomainResponse;
import
 software.amazon.awssdk.services.opensearch.model.DescribeDomainChangeProgressRequest;
import
 software.amazon.awssdk.services.opensearch.model.DescribeDomainChangeProgressResponse;
import software.amazon.awssdk.services.opensearch.model.DescribeDomainRequest;
import software.amazon.awssdk.services.opensearch.model.DomainInfo;
import software.amazon.awssdk.services.opensearch.model.DomainStatus;
import software.amazon.awssdk.services.opensearch.model.EBSOptions;
import software.amazon.awssdk.services.opensearch.model.ListDomainNamesRequest;
import software.amazon.awssdk.services.opensearch.model.ListTagsRequest;
import software.amazon.awssdk.services.opensearch.model.ListTagsResponse;
import software.amazon.awssdk.services.opensearch.model.NodeToNodeEncryptionOptions;
import software.amazon.awssdk.services.opensearch.model.Tag;
import software.amazon.awssdk.services.opensearch.model.UpdateDomainConfigRequest;
import software.amazon.awssdk.services.opensearch.model.UpdateDomainConfigResponse;
import software.amazon.awssdk.services.opensearch.model.VolumeType;
import java.time.Duration;
import java.util.ArrayList;
import java.util.List;
import java.util.concurrent.CompletableFuture;

public class OpenSearchActions {
    private static final Logger logger =
 LoggerFactory.getLogger(OpenSearchActions.class);
    private static OpenSearchAsyncClient openSearchClientAsyncClient;
    private static OpenSearchAsyncClient getAsyncClient() {
        if (openSearchClientAsyncClient == null) {
            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                .maxConcurrency(100)
                .connectionTimeout(Duration.ofSeconds(60))
                .readTimeout(Duration.ofSeconds(60))
                .writeTimeout(Duration.ofSeconds(60))
                .build();

            ClientOverrideConfiguration overrideConfig =
 ClientOverrideConfiguration.builder()
                .apiCallTimeout(Duration.ofMinutes(2))
                .apiCallAttemptTimeout(Duration.ofSeconds(90))
                .retryPolicy(RetryPolicy.builder()
                    .numRetries(3)
                    .build())
                .build();
```

```
            openSearchClientAsyncClient = OpenSearchAsyncClient.builder()
                .region(Region.US_EAST_1)
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return openSearchClientAsyncClient;
    }

    /**
     * Creates a new OpenSearch domain asynchronously.
     * @param domainName the name of the new OpenSearch domain to create
     * @return a {@link CompletableFuture} containing the domain ID of the newly
created domain
     */
    public CompletableFuture<String> createNewDomainAsync(String domainName) {
        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();
        logger.info("Sending domain creation request...");
        return getAsyncClient().createDomain(domainRequest)
```

```
                .handle( (createResponse, throwable) -> {
                    if (createResponse != null) {
                        logger.info("Domain status is {}",
createResponse.domainStatus().changeProgressDetails().configChangeStatusAsString());
                        logger.info("Domain Id is {}",
createResponse.domainStatus().domainId());
                        return createResponse.domainStatus().domainId();
                    }
                    throw new RuntimeException("Failed to create domain",
throwable);
                });
    }

    /**
     * Deletes a specific domain asynchronously.
     * @param domainName the name of the domain to be deleted
     * @return a {@link CompletableFuture} that completes when the domain has been
deleted
     * or throws a {@link RuntimeException} if the deletion fails
     */
    public CompletableFuture<DeleteDomainResponse> deleteSpecificDomainAsync(String
domainName) {
        DeleteDomainRequest domainRequest = DeleteDomainRequest.builder()
            .domainName(domainName)
            .build();

        // Delete domain asynchronously
        return getAsyncClient().deleteDomain(domainRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    throw new RuntimeException("Failed to delete the domain: " +
domainName, exception);
                }
            });
    }

    /**
     * Describes the specified domain asynchronously.
     *
     * @param domainName the name of the domain to describe
     * @return a {@link CompletableFuture} that completes with the ARN of the domain
     * @throws RuntimeException if the domain description fails
     */
    public CompletableFuture<String> describeDomainAsync(String domainName) {
```

```
        DescribeDomainRequest request = DescribeDomainRequest.builder()
            .domainName(domainName)
            .build();

        return getAsyncClient().describeDomain(request)
            .handle((response, exception) -> {  // Handle both response and
exception
                if (exception != null) {
                    throw new RuntimeException("Failed to describe domain",
exception);
                }
                DomainStatus domainStatus = response.domainStatus();
                String endpoint = domainStatus.endpoint();
                String arn = domainStatus.arn();
                String engineVersion = domainStatus.engineVersion();
                logger.info("Domain endpoint is: " + endpoint);
                logger.info("ARN: " + arn);
                System.out.println("Engine version: " + engineVersion);

                return arn;  // Return ARN when successful
            });
    }

    /**
     * Asynchronously lists all the domains in the current AWS account.
     * @return a {@link CompletableFuture} that, when completed, contains a list of
{@link DomainInfo} objects representing
     *         the domains in the account.
     * @throws RuntimeException if there was a failure while listing the domains.
     */
    public CompletableFuture<List<DomainInfo>> listAllDomainsAsync() {
        ListDomainNamesRequest namesRequest = ListDomainNamesRequest.builder()
            .engineType("OpenSearch")
            .build();

        return getAsyncClient().listDomainNames(namesRequest)
            .handle((response, exception) -> {
                if (exception != null) {
                    throw new RuntimeException("Failed to list all domains",
exception);
                }
                return response.domainNames();  // Return the list of domain names
on success
            });
```

```
    }

    /**
     * Updates the configuration of a specific domain asynchronously.
     * @param domainName the name of the domain to update
     * @return a {@link CompletableFuture} that represents the asynchronous
operation of updating the domain configuration
     */
    public CompletableFuture<UpdateDomainConfigResponse>
updateSpecificDomainAsync(String domainName) {
        ClusterConfig clusterConfig = ClusterConfig.builder()
            .instanceCount(3)
            .build();

        UpdateDomainConfigRequest updateDomainConfigRequest =
UpdateDomainConfigRequest.builder()
            .domainName(domainName)
            .clusterConfig(clusterConfig)
            .build();

        return getAsyncClient().updateDomainConfig(updateDomainConfigRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    throw new RuntimeException("Failed to update the domain
configuration", exception);
                }
                // Handle success if needed (e.g., logging or additional actions)
            });
    }

    /**
     * Asynchronously checks the progress of a domain change operation in Amazon
OpenSearch Service.
     * @param domainName the name of the OpenSearch domain to check the progress for
     * @return a {@link CompletableFuture} that completes when the domain change
operation is completed
     */
    public CompletableFuture<Void> domainChangeProgressAsync(String domainName) {
        DescribeDomainChangeProgressRequest request =
DescribeDomainChangeProgressRequest.builder()
            .domainName(domainName)
            .build();

        return CompletableFuture.runAsync(() -> {
```

```
            boolean isCompleted = false;
            long startTime = System.currentTimeMillis();

            while (!isCompleted) {
                try {
                    // Handle the async client call using `join` to block
synchronously for the result
                    DescribeDomainChangeProgressResponse response = getAsyncClient()
                        .describeDomainChangeProgress(request)
                        .handle((resp, ex) -> {
                            if (ex != null) {
                                throw new RuntimeException("Failed to check domain
progress", ex);
                            }
                            return resp;
                        }).join();

                    String state = response.changeProgressStatus().statusAsString();
 // Get the status as string

                    if ("COMPLETED".equals(state)) {
                        logger.info("\nOpenSearch domain status: Completed");
                        isCompleted = true;
                    } else {
                        for (int i = 0; i < 5; i++) {
                            long elapsedTimeInSeconds = (System.currentTimeMillis()
- startTime) / 1000;
                            String formattedTime = String.format("%02d:%02d",
elapsedTimeInSeconds / 60, elapsedTimeInSeconds % 60);
                            System.out.print("\rOpenSearch domain state: " + state +
" | Time Elapsed: " + formattedTime + " ");
                            System.out.flush();
                            Thread.sleep(1_000);
                        }
                    }
                } catch (InterruptedException e) {
                    Thread.currentThread().interrupt();
                    throw new RuntimeException("Thread was interrupted", e);
                }
            }
        });
    }

    /**
```

```
    * Asynchronously adds tags to an Amazon OpenSearch Service domain.
    * @param domainARN the Amazon Resource Name (ARN) of the Amazon OpenSearch
Service domain to add tags to
    * @return a {@link CompletableFuture} that completes when the tags have been
successfully added to the domain,
    * or throws a {@link RuntimeException} if the operation fails
    */
   public CompletableFuture<AddTagsResponse> addDomainTagsAsync(String domainARN) {
       Tag tag1 = Tag.builder()
           .key("service")
           .value("OpenSearch")
           .build();

       Tag tag2 = Tag.builder()
           .key("instances")
           .value("m3.2xlarge")
           .build();

       List<Tag> tagList = new ArrayList<>();
       tagList.add(tag1);
       tagList.add(tag2);

       AddTagsRequest addTagsRequest = AddTagsRequest.builder()
           .arn(domainARN)
           .tagList(tagList)
           .build();

       return getAsyncClient().addTags(addTagsRequest)
           .whenComplete((response, exception) -> {
               if (exception != null) {
                   throw new RuntimeException("Failed to add tags to the domain: "
+ domainARN, exception);
               } else {
                   logger.info("Added Tags");
               }
           });
   }


   /**
    * Asynchronously lists the tags associated with the specified Amazon Resource
Name (ARN).
    * @param arn the Amazon Resource Name (ARN) of the resource for which to list
the tags
```

```
     * @return a {@link CompletableFuture} that, when completed, will contain a list
 of the tags associated with the
     * specified ARN
     * @throws RuntimeException if there is an error listing the tags
     */
    public CompletableFuture<ListTagsResponse> listDomainTagsAsync(String arn) {
        ListTagsRequest tagsRequest = ListTagsRequest.builder()
            .arn(arn)
            .build();

        return getAsyncClient().listTags(tagsRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    throw new RuntimeException("Failed to list domain tags",
 exception);
                }

                List<Tag> tagList = response.tagList();
                for (Tag tag : tagList) {
                    logger.info("Tag key is " + tag.key());
                    logger.info("Tag value is " + tag.value());
                }
            });
    }
}
```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella Documentazione di riferimento delle API AWS SDK for Java 2.x .
  - AddTags
  - CreateDomain
  - DeleteDomain
  - DescribeDomain
  - DescribeDomainChangeProgress
  - ListDomainNames
  - ListTags
  - UpdateDomainConfig

Azioni

`AddTags`

Il seguente esempio di codice mostra come utilizzare. `AddTags`

SDK per Java 2.x

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
   /**
    * Asynchronously adds tags to an Amazon OpenSearch Service domain.
    * @param domainARN the Amazon Resource Name (ARN) of the Amazon OpenSearch
Service domain to add tags to
    * @return a {@link CompletableFuture} that completes when the tags have been
successfully added to the domain,
    * or throws a {@link RuntimeException} if the operation fails
    */
   public CompletableFuture<AddTagsResponse> addDomainTagsAsync(String domainARN) {
       Tag tag1 = Tag.builder()
           .key("service")
           .value("OpenSearch")
           .build();

       Tag tag2 = Tag.builder()
           .key("instances")
           .value("m3.2xlarge")
           .build();

       List<Tag> tagList = new ArrayList<>();
       tagList.add(tag1);
       tagList.add(tag2);

       AddTagsRequest addTagsRequest = AddTagsRequest.builder()
           .arn(domainARN)
           .tagList(tagList)
           .build();
```

```
        return getAsyncClient().addTags(addTagsRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    throw new RuntimeException("Failed to add tags to the domain: "
+ domainARN, exception);
                } else {
                    logger.info("Added Tags");
                }
            });
    }
```

- Per i dettagli sull'API, AddTagsconsulta AWS SDK for Java 2.x API Reference.

`ChangeProgress`

Il seguente esempio di codice mostra come utilizzare`ChangeProgress`.

SDK per Java 2.x

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
    /**
     * Asynchronously checks the progress of a domain change operation in Amazon
OpenSearch Service.
     * @param domainName the name of the OpenSearch domain to check the progress for
     * @return a {@link CompletableFuture} that completes when the domain change
operation is completed
     */
    public CompletableFuture<Void> domainChangeProgressAsync(String domainName) {
        DescribeDomainChangeProgressRequest request =
DescribeDomainChangeProgressRequest.builder()
            .domainName(domainName)
            .build();

        return CompletableFuture.runAsync(() -> {
            boolean isCompleted = false;
```

```
            long startTime = System.currentTimeMillis();

            while (!isCompleted) {
                try {
                    // Handle the async client call using `join` to block
synchronously for the result
                    DescribeDomainChangeProgressResponse response = getAsyncClient()
                        .describeDomainChangeProgress(request)
                        .handle((resp, ex) -> {
                            if (ex != null) {
                                throw new RuntimeException("Failed to check domain
progress", ex);
                            }
                            return resp;
                        }).join();

                    String state = response.changeProgressStatus().statusAsString();
 // Get the status as string

                    if ("COMPLETED".equals(state)) {
                        logger.info("\nOpenSearch domain status: Completed");
                        isCompleted = true;
                    } else {
                        for (int i = 0; i < 5; i++) {
                            long elapsedTimeInSeconds = (System.currentTimeMillis()
- startTime) / 1000;
                            String formattedTime = String.format("%02d:%02d",
elapsedTimeInSeconds / 60, elapsedTimeInSeconds % 60);
                            System.out.print("\rOpenSearch domain state: " + state +
" | Time Elapsed: " + formattedTime + " ");
                            System.out.flush();
                            Thread.sleep(1_000);
                        }
                    }
                } catch (InterruptedException e) {
                    Thread.currentThread().interrupt();
                    throw new RuntimeException("Thread was interrupted", e);
                }
            }
        });
    }
```

- Per i dettagli sull'API, ChangeProgressconsulta AWS SDK for Java 2.x API Reference.

`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.

```
   /**
    * Creates a new OpenSearch domain asynchronously.
    * @param domainName the name of the new OpenSearch domain to create
    * @return a {@link CompletableFuture} containing the domain ID of the newly
created domain
    */
   public CompletableFuture<String> createNewDomainAsync(String domainName) {
       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();
        logger.info("Sending domain creation request...");
        return getAsyncClient().createDomain(domainRequest)
                .handle( (createResponse, throwable) -> {
                    if (createResponse != null) {
                        logger.info("Domain status is {}",
createResponse.domainStatus().changeProgressDetails().configChangeStatusAsString());
                        logger.info("Domain Id is {}",
createResponse.domainStatus().domainId());
                        return createResponse.domainStatus().domainId();
                    }
                    throw new RuntimeException("Failed to create domain",
throwable);
                });
    }
```

- Per i dettagli sull'API, CreateDomainconsulta 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.

```
    /**
     * Deletes a specific domain asynchronously.
     * @param domainName the name of the domain to be deleted
     * @return a {@link CompletableFuture} that completes when the domain has been
deleted
     * or throws a {@link RuntimeException} if the deletion fails
     */
    public CompletableFuture<DeleteDomainResponse> deleteSpecificDomainAsync(String
domainName) {
        DeleteDomainRequest domainRequest = DeleteDomainRequest.builder()
```

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

        // Delete domain asynchronously
        return getAsyncClient().deleteDomain(domainRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    throw new RuntimeException("Failed to delete the domain: " +
domainName, exception);
                }
            });
    }
```

- Per i dettagli sull'API, DeleteDomainconsulta AWS SDK for Java 2.x API Reference.

`DescribeDomain`

Il seguente esempio di codice mostra come utilizzare`DescribeDomain`.

SDK per Java 2.x

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
    /**
     * Updates the configuration of a specific domain asynchronously.
     * @param domainName the name of the domain to update
     * @return a {@link CompletableFuture} that represents the asynchronous
operation of updating the domain configuration
     */
    public CompletableFuture<UpdateDomainConfigResponse>
updateSpecificDomainAsync(String domainName) {
        ClusterConfig clusterConfig = ClusterConfig.builder()
            .instanceCount(3)
            .build();

        UpdateDomainConfigRequest updateDomainConfigRequest =
UpdateDomainConfigRequest.builder()
```

```
            .domainName(domainName)
            .clusterConfig(clusterConfig)
            .build();

        return getAsyncClient().updateDomainConfig(updateDomainConfigRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    throw new RuntimeException("Failed to update the domain
configuration", exception);
                }
                // Handle success if needed (e.g., logging or additional actions)
            });
    }
```

- Per i dettagli sull'API, DescribeDomainconsulta 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.

```
    /**
     * Asynchronously lists all the domains in the current AWS account.
     * @return a {@link CompletableFuture} that, when completed, contains a list of
{@link DomainInfo} objects representing
     *         the domains in the account.
     * @throws RuntimeException if there was a failure while listing the domains.
     */
    public CompletableFuture<List<DomainInfo>> listAllDomainsAsync() {
        ListDomainNamesRequest namesRequest = ListDomainNamesRequest.builder()
            .engineType("OpenSearch")
            .build();
```

```
        return getAsyncClient().listDomainNames(namesRequest)
            .handle((response, exception) -> {
                if (exception != null) {
                    throw new RuntimeException("Failed to list all domains",
exception);
                }
                return response.domainNames();  // Return the list of domain names
on success
            });
    }
```

- Per i dettagli sull'API, ListDomainNamesconsulta AWS SDK for Java 2.x API Reference.

`ListTags`

Il seguente esempio di codice mostra come utilizzare`ListTags`.

SDK per Java 2.x

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
    /**
     * Asynchronously adds tags to an Amazon OpenSearch Service domain.
     * @param domainARN the Amazon Resource Name (ARN) of the Amazon OpenSearch
Service domain to add tags to
     * @return a {@link CompletableFuture} that completes when the tags have been
successfully added to the domain,
     * or throws a {@link RuntimeException} if the operation fails
     */
    public CompletableFuture<AddTagsResponse> addDomainTagsAsync(String domainARN) {
        Tag tag1 = Tag.builder()
            .key("service")
            .value("OpenSearch")
            .build();

        Tag tag2 = Tag.builder()
            .key("instances")
```

```
            .value("m3.2xlarge")
            .build();

        List<Tag> tagList = new ArrayList<>();
        tagList.add(tag1);
        tagList.add(tag2);

        AddTagsRequest addTagsRequest = AddTagsRequest.builder()
            .arn(domainARN)
            .tagList(tagList)
            .build();

        return getAsyncClient().addTags(addTagsRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    throw new RuntimeException("Failed to add tags to the domain: "
+ domainARN, exception);
                } else {
                    logger.info("Added Tags");
                }
            });
    }
```

- Per i dettagli sull'API, ListTagsconsulta 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.

```
    /**
     * Updates the configuration of a specific domain asynchronously.
     * @param domainName the name of the domain to update
```

```
     * @return a {@link CompletableFuture} that represents the asynchronous
operation of updating the domain configuration
     */
    public CompletableFuture<UpdateDomainConfigResponse>
updateSpecificDomainAsync(String domainName) {
        ClusterConfig clusterConfig = ClusterConfig.builder()
            .instanceCount(3)
            .build();

        UpdateDomainConfigRequest updateDomainConfigRequest =
UpdateDomainConfigRequest.builder()
            .domainName(domainName)
            .clusterConfig(clusterConfig)
            .build();

        return getAsyncClient().updateDomainConfig(updateDomainConfigRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    throw new RuntimeException("Failed to update the domain
configuration", exception);
                }
                // Handle success if needed (e.g., logging or additional actions)
            });
    }
```

- Per i dettagli sull'API, UpdateDomainConfigconsulta 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 nozioni di base sono esempi di codice che mostrano come eseguire le operazioni essenziali all'interno di un servizio.

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.

Gli scenari sono esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio o combinate con altri Servizi AWS.

Ogni esempio include un collegamento al codice sorgente completo, in cui è possibile trovare istruzioni su come configurare ed eseguire il codice nel contesto.

Nozioni di base

Salve EventBridge

L'esempio di codice seguente mostra come iniziare a utilizzare EventBridge.

SDK per Java 2.x

> **Note**
>
> C'è 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
 *
 */
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, ListEventBusesconsulta AWS SDK for Java 2.x API Reference.

Argomenti

- Nozioni di base
- Azioni
- Scenari

Nozioni di base

Informazioni di base

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.getAbsoluteFile());
        BufferedWriter bw = new BufferedWriter(fw);
        bw.write("This is a sample file for testing uploads.");
        bw.close();

        try {
            PutObjectRequest putOb = PutObjectRequest.builder()
                    .bucket(bucketName)
                    .key(fileName)
```

```
                    .build();

            s3Client.putObject(putOb, RequestBody.fromFile(myFile));

        } catch (S3Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void listTargetRules(EventBridgeClient eventBrClient, String
topicArn) {
        ListRuleNamesByTargetRequest ruleNamesByTargetRequest =
ListRuleNamesByTargetRequest.builder()
                .targetArn(topicArn)
                .build();

        ListRuleNamesByTargetResponse response =
eventBrClient.listRuleNamesByTarget(ruleNamesByTargetRequest);
        List<String> rules = response.ruleNames();
        for (String rule : rules) {
            System.out.println("The rule name is " + rule);
        }
    }

    public static void listTargets(EventBridgeClient eventBrClient, String ruleName)
{
        ListTargetsByRuleRequest ruleRequest = ListTargetsByRuleRequest.builder()
                .rule(ruleName)
                .build();

        ListTargetsByRuleResponse res =
eventBrClient.listTargetsByRule(ruleRequest);
        List<Target> targetsList = res.targets();
        for (Target target: targetsList) {
            System.out.println("Target ARN: "+target.arn());
        }
    }

    // Add a rule which triggers an SNS target when a file is uploaded to an S3
    // bucket.
    public static void addSnsEventRule(EventBridgeClient eventBrClient, String
ruleName, String topicArn,
            String topicName, String eventRuleName, String bucketName) {
```

```
        String targetID = java.util.UUID.randomUUID().toString();
        Target myTarget = Target.builder()
                .id(targetID)
                .arn(topicArn)
                .build();

        List<Target> targets = new ArrayList<>();
        targets.add(myTarget);
        PutTargetsRequest request = PutTargetsRequest.builder()
                .eventBusName(null)
                .targets(targets)
                .rule(ruleName)
                .build();

        eventBrClient.putTargets(request);
        System.out.println("Added event rule " + eventRuleName + " with Amazon SNS
 target " + topicName + " for bucket "
                + bucketName + ".");
    }

    public static void subEmail(SnsClient snsClient, String topicArn, String email)
 {
        try {
            SubscribeRequest request = SubscribeRequest.builder()
                    .protocol("email")
                    .endpoint(email)
                    .returnSubscriptionArn(true)
                    .topicArn(topicArn)
                    .build();

            SubscribeResponse result = snsClient.subscribe(request);
            System.out.println("Subscription ARN: " + result.subscriptionArn() + "\n
\n Status is "
                    + result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void listRules(EventBridgeClient eventBrClient) {
        try {
            ListRulesRequest rulesRequest = ListRulesRequest.builder()
```

```
                    .eventBusName("default")
                    .limit(10)
                    .build();

            ListRulesResponse response = eventBrClient.listRules(rulesRequest);
            List<Rule> rules = response.rules();
            for (Rule rule : rules) {
                System.out.println("The rule name is : " + rule.name());
                System.out.println("The rule description is : " +
rule.description());
                System.out.println("The rule state is : " + rule.stateAsString());
            }

        } catch (EventBridgeException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static String createSnsTopic(SnsClient snsClient, String topicName) {
        String topicPolicy = "{" +
                "\"Version\": \"2012-10-17\"," +
                "\"Statement\": [{" +
                "\"Sid\": \"EventBridgePublishTopic\"," +
                "\"Effect\": \"Allow\"," +
                "\"Principal\": {" +
                "\"Service\": \"events.amazonaws.com\"" +
                "}," +
                "\"Resource\": \"*\"," +
                "\"Action\": \"sns:Publish\"" +
                "}]" +
                "}";

        Map<String, String> topicAttributes = new HashMap<>();
        topicAttributes.put("Policy", topicPolicy);
        CreateTopicRequest topicRequest = CreateTopicRequest.builder()
                .name(topicName)
                .attributes(topicAttributes)
                .build();

        CreateTopicResponse response = snsClient.createTopic(topicRequest);
        System.out.println("Added topic " + topicName + " for email
subscriptions.");
        return response.topicArn();
```

```
    }

    // Create a new event rule that triggers when an Amazon S3 object is created in
    // a bucket.
    public static void addEventRule(EventBridgeClient eventBrClient, String roleArn,
String bucketName,
            String eventRuleName) {
        String pattern = "{\n" +
                "  \"source\": [\"aws.s3\"],\n" +
                "  \"detail-type\": [\"Object Created\"],\n" +
                "  \"detail\": {\n" +
                "    \"bucket\": {\n" +
                "      \"name\": [\"" + bucketName + "\"]\n" +
                "    }\n" +
                "  }\n" +
                "}";

        try {
            PutRuleRequest ruleRequest = PutRuleRequest.builder()
                    .description("Created by using the AWS SDK for Java v2")
                    .name(eventRuleName)
                    .eventPattern(pattern)
                    .roleArn(roleArn)
                    .build();

            PutRuleResponse ruleResponse = eventBrClient.putRule(ruleRequest);
            System.out.println("The ARN of the new rule is " +
ruleResponse.ruleArn());

        } catch (EventBridgeException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    // Determine if the S3 bucket exists.
    public static Boolean checkBucket(S3Client s3Client, String bucketName) {
        try {
            HeadBucketRequest headBucketRequest = HeadBucketRequest.builder()
                    .bucket(bucketName)
                    .build();

            s3Client.headBucket(headBucketRequest);
            return true;
```

```
        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }
        return false;
    }

    // Set the S3 bucket notification configuration.
    public static void setBucketNotification(S3Client s3Client, String bucketName) {
        try {
            EventBridgeConfiguration eventBridgeConfiguration =
EventBridgeConfiguration.builder()
                    .build();

            NotificationConfiguration configuration =
NotificationConfiguration.builder()
                    .eventBridgeConfiguration(eventBridgeConfiguration)
                    .build();

            PutBucketNotificationConfigurationRequest configurationRequest =
PutBucketNotificationConfigurationRequest
                    .builder()
                    .bucket(bucketName)
                    .notificationConfiguration(configuration)
                    .skipDestinationValidation(true)
                    .build();

            s3Client.putBucketNotificationConfiguration(configurationRequest);
            System.out.println("Added bucket " + bucketName + " with EventBridge
events enabled.");

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void createBucket(S3Client s3Client, String bucketName) {
        try {
            S3Waiter s3Waiter = s3Client.waiter();
            CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
                    .bucket(bucketName)
                    .build();

            s3Client.createBucket(bucketRequest);
```

```
            HeadBucketRequest bucketRequestWait = HeadBucketRequest.builder()
                    .bucket(bucketName)
                    .build();

            // Wait until the bucket is created and print out the response.
            WaiterResponse<HeadBucketResponse> waiterResponse =
 s3Waiter.waitUntilBucketExists(bucketRequestWait);
            waiterResponse.matched().response().ifPresent(System.out::println);
            System.out.println(bucketName + " is ready");

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static String createIAMRole(IamClient iam, String rolename, String
 polJSON) {
        try {
            CreateRoleRequest request = CreateRoleRequest.builder()
                    .roleName(rolename)
                    .assumeRolePolicyDocument(polJSON)
                    .description("Created using the AWS SDK for Java")
                    .build();

            CreateRoleResponse response = iam.createRole(request);
            AttachRolePolicyRequest rolePolicyRequest =
 AttachRolePolicyRequest.builder()
                    .roleName(rolename)
                    .policyArn("arn:aws:iam::aws:policy/
AmazonEventBridgeFullAccess")
                    .build();

            iam.attachRolePolicy(rolePolicyRequest);
            return response.role().arn();

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
}
```

- 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

Azioni

`DeleteRule`

Il seguente esempio di codice mostra come usare`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, DeleteRuleconsulta 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, DescribeRuleconsulta 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, DisableRuleconsulta 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, EnableRuleconsulta 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, ListRuleNamesByTargetconsulta 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, ListRulesconsulta 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, ListTargetsByRuleconsulta 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, PutEventsconsulta 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, PutRuleconsulta 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, PutTargetsconsulta 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, RemoveTargetsconsulta AWS SDK for Java 2.x API Reference.

Scenari

Invia notifiche di eventi a EventBridge

Il seguente esempio di codice mostra come abilitare un bucket per inviare notifiche di eventi S3 EventBridge e indirizzarle verso un argomento Amazon SNS e una coda Amazon SQS.

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.

```
   /** This method configures a bucket to send events to AWS EventBridge and
creates a rule
    * to route the S3 object created events to a topic and a queue.
    *
    * @param bucketName Name of existing bucket
    * @param topicArn ARN of existing topic to receive S3 event notifications
    * @param queueArn ARN of existing queue to receive S3 event notifications
    *
    *  An AWS CloudFormation stack sets up the bucket, queue, topic before the
method runs.
    */
   public static String setBucketNotificationToEventBridge(String bucketName,
String topicArn, String queueArn) {
       try {
           // Enable bucket to emit S3 Event notifications to EventBridge.
           s3Client.putBucketNotificationConfiguration(b -> b
                   .bucket(bucketName)
                   .notificationConfiguration(b1 -> b1
                           .eventBridgeConfiguration(
                                   SdkBuilder::build)
                   ).build()).join();

           // Create an EventBridge rule to route Object Created notifications.
           PutRuleRequest putRuleRequest = PutRuleRequest.builder()
                   .name(RULE_NAME)
                   .eventPattern("""
                           {
                             "source": ["aws.s3"],
                             "detail-type": ["Object Created"],
                             "detail": {
                               "bucket": {
                                 "name": ["%s"]
                               }
                             }
```

```
                            }
                            """.formatted(bucketName))
                    .build();

            // Add the rule to the default event bus.
            PutRuleResponse putRuleResponse =
eventBridgeClient.putRule(putRuleRequest)
                    .whenComplete((r, t) -> {
                        if (t != null) {
                            logger.error("Error creating event bus rule: " +
t.getMessage(), t);
                            throw new RuntimeException(t.getCause().getMessage(),
t);
                        }
                        logger.info("Event bus rule creation request sent
successfully. ARN is: {}", r.ruleArn());
                    }).join();

            // Add the existing SNS topic and SQS queue as targets to the rule.
            eventBridgeClient.putTargets(b -> b
                    .eventBusName("default")
                    .rule(RULE_NAME)
                    .targets(List.of (
                            Target.builder()
                                    .arn(queueArn)
                                    .id("Queue")
                                    .build(),
                            Target.builder()
                                    .arn(topicArn)
                                    .id("Topic")
                                    .build())
                            )
                    ).join();
            return putRuleResponse.ruleArn();
        } catch (S3Exception 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 .
  - PutBucketNotificationConfiguration
  - PutRule
  - PutTargets

Utilizzo degli eventi pianificati per richiamare una funzione Lambda

Il seguente esempio di codice mostra come creare una AWS Lambda funzione richiamata da un evento EventBridge pianificato di Amazon.

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

- CloudWatch Registri
- DynamoDB
- EventBridge
- Lambda
- Amazon SNS

EventBridge Esempi di scheduler 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 Scheduler.

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.

Gli scenari sono esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio o combinate con altri Servizi AWS.

Ogni esempio include un collegamento al codice sorgente completo, in cui è possibile trovare istruzioni su come configurare ed eseguire il codice nel contesto.

Nozioni di base

Ciao EventBridge Scheduler

I seguenti esempi di codice mostrano come iniziare a utilizzare EventBridge Scheduler.

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.services.scheduler.SchedulerAsyncClient;
import software.amazon.awssdk.services.scheduler.model.ListSchedulesRequest;
import software.amazon.awssdk.services.scheduler.model.ScheduleSummary;
import software.amazon.awssdk.services.scheduler.paginators.ListSchedulesPublisher;
import java.util.ArrayList;
import java.util.List;
import java.util.concurrent.CompletableFuture;

public class HelloScheduler {

    public static void main(String [] args) {
        listSchedulesAsync();
    }

    /**
     * Lists all the schedules available.
     * <p>
```

```
    * This method uses the {@link SchedulerAsyncClient} to make an asynchronous
request to
    * list all the schedules available. The method uses the {@link
ListSchedulesPublisher}
    * to fetch the schedules in a paginated manner, and then processes the
responses
    * asynchronously.
    */
   public static void listSchedulesAsync() {
       SchedulerAsyncClient schedulerAsyncClient = SchedulerAsyncClient.create();

       // Build the request to list schedules
       ListSchedulesRequest listSchedulesRequest =
ListSchedulesRequest.builder().build();

       // Use the paginator to fetch all schedules asynchronously.
       ListSchedulesPublisher paginator =
schedulerAsyncClient.listSchedulesPaginator(listSchedulesRequest);
       List<ScheduleSummary> results = new ArrayList<>();

       // Subscribe to the paginator to process the response asynchronously
       CompletableFuture<Void> future = paginator.subscribe(response -> {
           response.schedules().forEach(schedule -> {
               results.add(schedule);
               System.out.printf("Schedule: %s%n", schedule.name());
           });
       });

       // Wait for the asynchronous operation to complete.
       future.join();

       // After all schedules are fetched, print the total count.
       System.out.printf("Total of %d schedule(s) available.%n", results.size());
   }
}
```

- Per i dettagli sull'API, ListSchedulesconsulta AWS SDK for Java 2.x API Reference.

Argomenti

- Azioni
- Scenari

Azioni

`CreateSchedule`

Il seguente esempio di codice mostra come utilizzare`CreateSchedule`.

SDK per Java 2.x

> **Note**
>
> C'è 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 a new schedule for a target task.
    *
    * @param name                  the name of the schedule
    * @param scheduleExpression    The schedule expression that defines when the
schedule should run.
    * @param scheduleGroupName     the name of the schedule group to which the
schedule belongs
    * @param targetArn             the Amazon Resource Name (ARN) of the target
task
    * @param roleArn               the ARN of the IAM role to be used for the
schedule
    * @param input                 the input data for the target task
    * @param deleteAfterCompletion whether to delete the schedule after it's
executed
    * @param useFlexibleTimeWindow whether to use a flexible time window for the
schedule execution
    * @return true if the schedule was successfully created, false otherwise
    */
   public CompletableFuture<Boolean> createScheduleAsync(
       String name,
       String scheduleExpression,
       String scheduleGroupName,
       String targetArn,
       String roleArn,
       String input,
       boolean deleteAfterCompletion,
```

```
        boolean useFlexibleTimeWindow) {

        int hoursToRun = 1;
        int flexibleTimeWindowMinutes = 10;

        Target target = Target.builder()
            .arn(targetArn)
            .roleArn(roleArn)
            .input(input)
            .build();

        FlexibleTimeWindow flexibleTimeWindow = FlexibleTimeWindow.builder()
            .mode(useFlexibleTimeWindow
                ? FlexibleTimeWindowMode.FLEXIBLE
                : FlexibleTimeWindowMode.OFF)
            .maximumWindowInMinutes(useFlexibleTimeWindow
                ? flexibleTimeWindowMinutes
                : null)
            .build();

        Instant startDate = Instant.now();
        Instant endDate = startDate.plus(Duration.ofHours(hoursToRun));

        CreateScheduleRequest request = CreateScheduleRequest.builder()
            .name(name)
            .scheduleExpression(scheduleExpression)
            .groupName(scheduleGroupName)
            .target(target)
            .actionAfterCompletion(deleteAfterCompletion
                ? ActionAfterCompletion.DELETE
                : ActionAfterCompletion.NONE)
            .startDate(startDate)
            .endDate(endDate)
            .flexibleTimeWindow(flexibleTimeWindow)
            .build();

        return getAsyncClient().createSchedule(request)
            .thenApply(response -> {
                logger.info("Successfully created schedule {} in schedule group {},
The ARN is {} ", name, scheduleGroupName, response.scheduleArn());
                return true;
            })
            .whenComplete((result, ex) -> {
                if (ex != null) {
```

```
                    if (ex instanceof ConflictException) {
                        // Handle ConflictException
                        logger.error("A conflict exception occurred while creating
the schedule: {}", ex.getMessage());
                        throw new CompletionException("A conflict exception occurred
while creating the schedule: " + ex.getMessage(), ex);
                    } else {
                        throw new CompletionException("Error creating schedule: " +
ex.getMessage(), ex);
                    }
                }
            });
    }
```

- Per i dettagli sull'API, CreateScheduleconsulta AWS SDK for Java 2.x API Reference.

`CreateScheduleGroup`

Il seguente esempio di codice mostra come utilizzare`CreateScheduleGroup`.

SDK per Java 2.x

> **Note**
>
> C'è 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 a new schedule group.
     *
     * @param name the name of the schedule group to be created
     * @return a {@link CompletableFuture} representing the asynchronous operation
of creating the schedule group
     */
    public CompletableFuture<CreateScheduleGroupResponse> createScheduleGroup(String
name) {
        CreateScheduleGroupRequest request = CreateScheduleGroupRequest.builder()
            .name(name)
            .build();
```

```
        logger.info("Initiating createScheduleGroup call for group: {}", name);
        CompletableFuture<CreateScheduleGroupResponse> futureResponse =
getAsyncClient().createScheduleGroup(request);
        futureResponse.whenComplete((response, ex) -> {
            if (ex != null) {
                if (ex instanceof CompletionException && ex.getCause() instanceof
ConflictException) {
                    // Rethrow the ConflictException
                    throw (ConflictException) ex.getCause();
                } else {
                    throw new CompletionException("Failed to create schedule group:
" + name, ex);
                }
            } else if (response == null) {
                throw new RuntimeException("Failed to create schedule group:
response was null");
            } else {
                logger.info("Successfully created schedule group '{}': {}", name,
response.scheduleGroupArn());
            }
        });

        return futureResponse;
    }
```

- Per i dettagli sull'API, CreateScheduleGroupconsulta AWS SDK for Java 2.x API Reference.

`DeleteSchedule`

Il seguente esempio di codice mostra come utilizzare`DeleteSchedule`.

SDK per Java 2.x

> **Note**
>
> C'è 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 schedule with the specified name and group name.
    *
    * @param name      the name of the schedule to be deleted
    * @param groupName the group name of the schedule to be deleted
    * @return a {@link CompletableFuture} that, when completed, indicates whether
the schedule was successfully deleted
    * @throws CompletionException if an error occurs while deleting the schedule,
except for the case where the schedule is not found
    */
   public CompletableFuture<Boolean> deleteScheduleAsync(String name, String
groupName) {
       DeleteScheduleRequest request = DeleteScheduleRequest.builder()
           .name(name)
           .groupName(groupName)
           .build();

       CompletableFuture<DeleteScheduleResponse> response =
getAsyncClient().deleteSchedule(request);
       return response.handle((result, ex) -> {
           if (ex != null) {
               if (ex instanceof ResourceNotFoundException) {
                   throw new CompletionException("Resource not found while deleting
schedule with ID: " + name, ex);
               } else {
                   throw new CompletionException("Failed to delete schedule.", ex);
               }
           }
           logger.info("Successfully deleted schedule with name {}.", name);
           return true;
       });
   }
```

- Per i dettagli sull'API, DeleteScheduleconsulta AWS SDK for Java 2.x API Reference.

`DeleteScheduleGroup`

Il seguente esempio di codice mostra come utilizzare`DeleteScheduleGroup`.

SDK per Java 2.x

> **Note**
>
> C'è 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 specified schedule group.
    *
    * @param name the name of the schedule group to delete
    * @return a {@link CompletableFuture} that completes when the schedule group
has been deleted
    * @throws CompletionException if an error occurs while deleting the schedule
group
    */
   public CompletableFuture<Void> deleteScheduleGroupAsync(String name) {
       DeleteScheduleGroupRequest request = DeleteScheduleGroupRequest.builder()
           .name(name)
           .build();

       return getAsyncClient().deleteScheduleGroup(request)
           .thenRun(() -> {
               logger.info("Successfully deleted schedule group {}", name);
           })
           .whenComplete((result, ex) -> {
               if (ex != null) {
                   if (ex instanceof ResourceNotFoundException) {
                       throw new CompletionException("The resource was not found: "
+ ex.getMessage(), ex);
                   } else {
                       throw new CompletionException("Error deleting schedule
group: " + ex.getMessage(), ex);
                   }
               }
           });
   }
```

- Per i dettagli sull'API, DeleteScheduleGroupconsulta AWS SDK for Java 2.x API Reference.

Scenari

Eventi pianificati

L'esempio di codice seguente mostra come:

- Implementa uno AWS CloudFormation stack con le risorse richieste.
- Crea un gruppo di EventBridge pianificazione di Scheduler.
- Crea una EventBridge pianificazione di Scheduler una tantum con una finestra temporale flessibile.
- Crea una pianificazione ricorrente dello EventBridge Scheduler con una frequenza specificata.
- Elimina EventBridge Scheduler, la pianificazione e il gruppo di pianificazioni.
- Pulisci le risorse ed elimina lo stack.

SDK per Java 2.x

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

Esegui lo scenario.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.scheduler.model.SchedulerException;
import javax.mail.internet.AddressException;
import javax.mail.internet.InternetAddress;
import java.time.LocalDateTime;
import java.time.format.DateTimeFormatter;
import java.util.Map;
import java.util.Scanner;
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.CompletionException;

/**
 * This Java code example performs the following tasks for the Amazon EventBridge
 Scheduler workflow:
 * <p>
 * 1. Prepare the Application:
```

```
 * - Prompt the user for an email address to use for the subscription for the SNS
 topic subscription.
 * - Deploy the Cloud Formation template in resources/cfn_template.yaml for resource
 creation.
 * - Store the outputs of the stack into variables for use in the workflow.
 * - Create a schedule group for all workflow schedules.
 * <p>
 * 2. Create one-time Schedule:
 * - Create a one-time schedule to send an initial event.
 * - Use a Flexible Time Window and set the schedule to delete after completion.
 * - Wait for the user to receive the event email from SNS.
 * <p>
 * 3. Create a time-based schedule:
 * - Prompt the user for how many X times per Y hours a recurring event should be
 scheduled.
 * - Create the scheduled event for X times per hour for Y hours.
 * - Wait for the user to receive the event email from SNS.
 * - Delete the schedule when the user is finished.
 * <p>
 * 4. Clean up:
 * - Prompt the user for y/n answer if they want to destroy the stack and clean up
 all resources.
 * - Delete the schedule group.
 * - Destroy the Cloud Formation stack and wait until the stack has been removed.
 */

public class EventbridgeSchedulerScenario {

    private static final Logger logger =
 LoggerFactory.getLogger(EventbridgeSchedulerScenario.class);
    private static final Scanner scanner = new Scanner(System.in);
    private static String STACK_NAME = "workflow-stack-name";
    private static final String scheduleGroupName = "schedules-group";

    private static String recurringScheduleName = "";

    private static String oneTimeScheduleName = "";

    private static final EventbridgeSchedulerActions eventbridgeActions = new
 EventbridgeSchedulerActions();

    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    public static String roleArn = "";
```

```
   public static String snsTopicArn = "";

   public static void main(String[] args) {
       logger.info(DASHES);
       logger.info("Welcome to the Amazon EventBridge Scheduler Workflow.");
       logger.info("""
           Amazon EventBridge Scheduler is a fully managed service that helps you
schedule and execute
           a wide range of tasks and events in the cloud. It's designed to simplify
the process of
           scheduling and managing recurring or one-time events, making it easier
for developers and
           businesses to automate various workflows and processes.

           One of the key features of Amazon EventBridge Scheduler is its ability
to schedule events
           based on a variety of triggers, including time-based schedules, custom
event patterns, or
           even integration with other AWS services. For example, you can use
EventBridge Scheduler
           to schedule a report generation task to run every weekday at 9 AM, or to
trigger a
           Lambda function when a specific Amazon S3 object is created.

           This flexibility allows you to build complex and dynamic event-driven
architectures
           that adapt to your business needs.

           Lets get started...
           """);
       waitForInputToContinue();
       logger.info(DASHES);

       logger.info(DASHES);
       logger.info("1. Prepare the application.");
       waitForInputToContinue();
       try {
           boolean prepareSuccess = prepareApplication();
           logger.info(DASHES);

           if (prepareSuccess) {
               logger.info("2. Create one-time schedule.");
               logger.info("""
```

```
                    A one-time schedule in Amazon EventBridge Scheduler is an event
trigger that allows
                    you to schedule a one-time event to run at a specific date and
time. This is useful for
                    executing a specific task or workflow at a predetermined time,
without the need for recurring
                    or complex scheduling.
                    """);
                waitForInputToContinue();
                createOneTimeSchedule();
                logger.info("Do you want to delete the schedule {} (y/n) ?",
oneTimeScheduleName);
                String ans = scanner.nextLine().trim();
                if (ans.equalsIgnoreCase("y")) {

eventbridgeActions.deleteScheduleAsync(oneTimeScheduleName,scheduleGroupName);
                }
                logger.info(DASHES);

                logger.info("3. Create a recurring schedule.");
                logger.info("""
                    A recurring schedule is a feature that allows you to schedule
and manage the execution
                    of your serverless applications or workloads on a recurring
basis. For example,
                    with EventBridge Scheduler, you can create custom schedules for
your AWS Lambda functions,
                    AWS Step Functions, and other supported event sources, enabling
you to automate tasks and
                    workflows without the need for complex infrastructure
management.
                    """);
                waitForInputToContinue();
                createRecurringSchedule();
                logger.info("Do you want to delete the schedule {} (y/n) ?",
oneTimeScheduleName);
                String ans2 = scanner.nextLine().trim();
                if (ans2.equalsIgnoreCase("y")) {

eventbridgeActions.deleteScheduleAsync(recurringScheduleName,scheduleGroupName);
                }
                logger.info(DASHES);
            }
        } catch (Exception ex) {
```

```
            logger.info("There was a problem with the workflow {}, initiating
cleanup...", ex.getMessage());
            cleanUp();
        }

        logger.info(DASHES);
        logger.info("4. Clean up the resources.");
        logger.info("Do you want to delete these AWS resources (y/n) ?");
        String delAns = scanner.nextLine().trim();
        if (delAns.equalsIgnoreCase("y")) {
            cleanUp();
        } else {
            logger.info("The AWS resources will not be deleted.");
        }
        logger.info("Amazon EventBridge Scheduler workflow completed.");
        logger.info(DASHES);
    }

    /**
     * Cleans up the resources associated with the EventBridge scheduler.
     * If any errors occur during the cleanup process, the corresponding error
messages are logged.
     */
    public static void cleanUp() {
        logger.info("First, delete the schedule group.");
        logger.info("When the schedule group is deleted, schedules that are part of
that group are deleted.");
        waitForInputToContinue();
        try {
            eventbridgeActions.deleteScheduleGroupAsync(scheduleGroupName).join();

        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof SchedulerException schedulerException) {
                logger.error("Scheduler error occurred: Error message: {}, Error
code {}",
                    schedulerException.getMessage(),
schedulerException.awsErrorDetails().errorCode(), schedulerException);
            } else {
                logger.error("An unexpected error occurred: {}",
cause.getMessage());
            }
            return;
        }
```

```
        logger.info("Destroy the CloudFormation stack");
        waitForInputToContinue();
        CloudFormationHelper.destroyCloudFormationStack(STACK_NAME);
    }

    /**
     * Prepares the application by creating resources in a CloudFormation stack,
including an SNS topic
     * that will be subscribed to the EventBridge Scheduler events. The user will
need to confirm the subscription
     * in order to receive event emails.
     *
     * @return true if the application preparation was successful, false otherwise
     */
    public static boolean prepareApplication() {
        logger.info("""
            This example creates resources in a CloudFormation stack, including an
SNS topic
            that will be subscribed to the EventBridge Scheduler events.
            You will need to confirm the subscription in order to receive event
emails.
             """);

        String emailAddress = promptUserForEmail();
        logger.info("You entered {}", emailAddress);

        logger.info("Do you want to use a custom Stack name (y/n) ?");
        String ans = scanner.nextLine().trim();
        if (ans.equalsIgnoreCase("y")) {
            String newStackName = scanner.nextLine();
            logger.info("You entered {} for the new stack name", newStackName);
            waitForInputToContinue();
            STACK_NAME = newStackName;
        }

        logger.info("Get the roleArn and snsTopicArn values using a Cloudformation
template.");
        waitForInputToContinue();
        CloudFormationHelper.deployCloudFormationStack(STACK_NAME, emailAddress);
        Map<String, String> stackOutputs =
CloudFormationHelper.getStackOutputs(STACK_NAME);
        roleArn = stackOutputs.get("RoleARN");
        snsTopicArn = stackOutputs.get("SNStopicARN");
```

```
        logger.info("The roleARN is {}", roleArn);
        logger.info("The snsTopicArn is {}", snsTopicArn);

        try {
            eventbridgeActions.createScheduleGroup(scheduleGroupName).join();
            logger.info("createScheduleGroupAsync completed successfully.");

        } catch (RuntimeException e) {
            logger.error("Error occurred: {} ", e.getMessage());
            return false;
        }
        logger.info("Application preparation complete.");
        return true;
    }

    /**
     * Waits for the user to enter 'c' followed by <ENTER> to continue the program.
     * This method is used to pause the program execution and wait for user input
before
     * proceeding.
     */
    private static void waitForInputToContinue() {
        while (true) {
            logger.info("");
            logger.info("Enter 'c' followed by <ENTER> to continue:");
            String input = scanner.nextLine();

            if (input.trim().equalsIgnoreCase("c")) {
                logger.info("Continuing with the program...");
                logger.info("");
                break;
            } else {
                // Handle invalid input.
                logger.info("Invalid input. Please try again.");
            }
        }
    }

    /**
     * Prompts the user to enter an email address and validates the input.
     * If the provided email address is invalid, the method will prompt the user to
try again.
     *
```

```
     * @return the valid email address entered by the user
     */
    private static String promptUserForEmail() {
        logger.info("Enter an email address to use for event subscriptions: ");
        String email = scanner.nextLine();
        if (!isValidEmail(email)) {
            logger.info("Invalid email address. Please try again.");
            return promptUserForEmail();
        }
        return email;
    }

    /**
     * Checks if the given email address is valid.
     *
     * @param email the email address to be validated
     * @return {@code true} if the email address is valid, {@code false} otherwise
     */
    private static boolean isValidEmail(String email) {
        try {
            InternetAddress emailAddress = new InternetAddress(email);
            emailAddress.validate();
            return true;

        } catch (AddressException e) {
            return false;
        }
    }

    /**
     * Creates a one-time schedule to send an initial event in 1 minute with a
 flexible time window.
     *
     * @return {@code true} if the schedule was created successfully, {@code false}
 otherwise
     */
    public static Boolean createOneTimeSchedule() {
        oneTimeScheduleName = promptUserForResourceName("Enter a name for the one-
time schedule:");
        logger.info("Creating a one-time schedule named {} to send an initial event
 in 1 minute with a flexible time window...", oneTimeScheduleName);
        LocalDateTime scheduledTime = LocalDateTime.now();
        DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-
dd'T'HH:mm:ss");
```

```
        String scheduleExpression = "at(" + scheduledTime.format(formatter) + ")";
        return eventbridgeActions.createScheduleAsync(
            oneTimeScheduleName,
            scheduleExpression,
            scheduleGroupName,
            snsTopicArn,
            roleArn,
            "One time scheduled event test from schedule",
            true,
            true).join();
    }


    /**
     * Creates a recurring schedule to send events based on a specific time.
     *
     * @return A {@link CompletableFuture} that completes with a boolean value
indicating the success or failure of the operation.
     */
    public static Boolean createRecurringSchedule() {
        logger.info("Creating a recurring schedule to send events for one hour...");
        recurringScheduleName = promptUserForResourceName("Enter a name for the
recurring schedule:");

        // Prompt the user for the schedule rate (in minutes).
        int scheduleRateInMinutes = promptUserForInteger("Enter the desired schedule
rate (in minutes): ");
        String scheduleExpression = "rate(" + scheduleRateInMinutes + " minutes)";
        return eventbridgeActions.createScheduleAsync(
            recurringScheduleName,
            scheduleExpression,
            scheduleGroupName,
            snsTopicArn,
            roleArn,
            "Recurrent event test from schedule " + recurringScheduleName,
            true,
            true).join();
    }

    /**
     * Prompts the user for a resource name and validates the input.
     *
```

```
     * @param prompt the message to display to the user when prompting for the
 resource name
     * @return the valid resource name entered by the user
     */
    private static String promptUserForResourceName(String prompt) {
        logger.info(prompt);
        String resourceName = scanner.nextLine();
        String regex = "[0-9a-zA-Z-_.]+";
        if (!resourceName.matches(regex)) {
            logger.info("Invalid resource name. Please use a name that matches the
 pattern " + regex + ".");
            return promptUserForResourceName(prompt);
        }
        return resourceName;
    }

    /**
     * Prompts the user for an integer input and returns the integer value.
     *
     * @param prompt the message to be displayed to the user when prompting for
 input
     * @return the integer value entered by the user
     */
    private static int promptUserForInteger(String prompt) {
        logger.info(prompt);
        String stringResponse = scanner.nextLine();
        if (stringResponse == null || stringResponse.trim().isEmpty() || !
isInteger(stringResponse)) {
            logger.info("Invalid integer.");
            return promptUserForInteger(prompt);
        }
        return Integer.parseInt(stringResponse);
    }

    /**
     * Checks if the given string represents a valid integer.
     *
     * @param str the string to be checked
     * @return {@code true} if the string represents a valid integer, {@code false}
 otherwise
     */
    private static boolean isInteger(String str) {
        try {
            Integer.parseInt(str);
```

```
            return true;
        } catch (NumberFormatException e) {
            return false;
        }
    }
}
```

Wrapper per operazioni di assistenza.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.client.config.ClientOverrideConfiguration;
import software.amazon.awssdk.core.retry.RetryMode;
import software.amazon.awssdk.http.async.SdkAsyncHttpClient;
import software.amazon.awssdk.http.nio.netty.NettyNioAsyncHttpClient;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.scheduler.SchedulerAsyncClient;
import software.amazon.awssdk.services.scheduler.model.ActionAfterCompletion;
import software.amazon.awssdk.services.scheduler.model.ConflictException;
import software.amazon.awssdk.services.scheduler.model.CreateScheduleGroupRequest;
import software.amazon.awssdk.services.scheduler.model.CreateScheduleGroupResponse;
import software.amazon.awssdk.services.scheduler.model.CreateScheduleRequest;
import software.amazon.awssdk.services.scheduler.model.DeleteScheduleGroupRequest;
import software.amazon.awssdk.services.scheduler.model.DeleteScheduleRequest;
import software.amazon.awssdk.services.scheduler.model.DeleteScheduleResponse;
import software.amazon.awssdk.services.scheduler.model.FlexibleTimeWindow;
import software.amazon.awssdk.services.scheduler.model.FlexibleTimeWindowMode;
import software.amazon.awssdk.services.scheduler.model.ResourceNotFoundException;
import software.amazon.awssdk.services.scheduler.model.Target;

import java.time.Instant;
import java.util.concurrent.CompletableFuture;
import java.time.Duration;
import java.util.concurrent.CompletionException;

public class EventbridgeSchedulerActions {

    private static SchedulerAsyncClient schedulerClient;
    private static final Logger logger =
 LoggerFactory.getLogger(EventbridgeSchedulerActions.class);
```

```
   public static SchedulerAsyncClient getAsyncClient() {
       if (schedulerClient == null) {
           /*
           The `NettyNioAsyncHttpClient` class is part of the AWS SDK for Java,
version 2,
           and it is designed to provide a high-performance, asynchronous HTTP
client for interacting with AWS services.
            It uses the Netty framework to handle the underlying network
communication and the Java NIO API to
            provide a non-blocking, event-driven approach to HTTP requests and
responses.
            */

           SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
               .maxConcurrency(50)  // Adjust as needed.
               .connectionTimeout(Duration.ofSeconds(60))  // Set the connection
timeout.
               .readTimeout(Duration.ofSeconds(60))  // Set the read timeout.
               .writeTimeout(Duration.ofSeconds(60))  // Set the write timeout.
               .build();

           ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
               .apiCallTimeout(Duration.ofMinutes(2))  // Set the overall API call
timeout.
               .apiCallAttemptTimeout(Duration.ofSeconds(90))  // Set the
individual call attempt timeout.
               .retryStrategy(RetryMode.STANDARD)
               .build();

           schedulerClient = SchedulerAsyncClient.builder()
               .region(Region.US_EAST_1)
               .httpClient(httpClient)
               .overrideConfiguration(overrideConfig)
               .build();
       }
       return schedulerClient;
   }


   /**
    * Creates a new schedule group.
    *
    * @param name the name of the schedule group to be created
```

```
    * @return a {@link CompletableFuture} representing the asynchronous operation
of creating the schedule group
    */
   public CompletableFuture<CreateScheduleGroupResponse> createScheduleGroup(String
name) {
       CreateScheduleGroupRequest request = CreateScheduleGroupRequest.builder()
           .name(name)
           .build();

       logger.info("Initiating createScheduleGroup call for group: {}", name);
       CompletableFuture<CreateScheduleGroupResponse> futureResponse =
getAsyncClient().createScheduleGroup(request);
       futureResponse.whenComplete((response, ex) -> {
           if (ex != null) {
               if (ex instanceof CompletionException && ex.getCause() instanceof
ConflictException) {
                   // Rethrow the ConflictException
                   throw (ConflictException) ex.getCause();
               } else {
                   throw new CompletionException("Failed to create schedule group:
" + name, ex);
               }
           } else if (response == null) {
               throw new RuntimeException("Failed to create schedule group:
response was null");
           } else {
               logger.info("Successfully created schedule group '{}': {}", name,
response.scheduleGroupArn());
           }
       });

       return futureResponse;
   }


   /**
    * Creates a new schedule for a target task.
    *
    * @param name                  the name of the schedule
    * @param scheduleExpression    The schedule expression that defines when the
schedule should run.
    * @param scheduleGroupName     the name of the schedule group to which the
schedule belongs
```

```
    * @param targetArn             the Amazon Resource Name (ARN) of the target
task
    * @param roleArn               the ARN of the IAM role to be used for the
schedule
    * @param input                 the input data for the target task
    * @param deleteAfterCompletion whether to delete the schedule after it's
executed
    * @param useFlexibleTimeWindow whether to use a flexible time window for the
schedule execution
    * @return true if the schedule was successfully created, false otherwise
    */
   public CompletableFuture<Boolean> createScheduleAsync(
       String name,
       String scheduleExpression,
       String scheduleGroupName,
       String targetArn,
       String roleArn,
       String input,
       boolean deleteAfterCompletion,
       boolean useFlexibleTimeWindow) {

       int hoursToRun = 1;
       int flexibleTimeWindowMinutes = 10;

       Target target = Target.builder()
           .arn(targetArn)
           .roleArn(roleArn)
           .input(input)
           .build();

       FlexibleTimeWindow flexibleTimeWindow = FlexibleTimeWindow.builder()
           .mode(useFlexibleTimeWindow
               ? FlexibleTimeWindowMode.FLEXIBLE
               : FlexibleTimeWindowMode.OFF)
           .maximumWindowInMinutes(useFlexibleTimeWindow
               ? flexibleTimeWindowMinutes
               : null)
           .build();

       Instant startDate = Instant.now();
       Instant endDate = startDate.plus(Duration.ofHours(hoursToRun));

       CreateScheduleRequest request = CreateScheduleRequest.builder()
           .name(name)
```

```
            .scheduleExpression(scheduleExpression)
            .groupName(scheduleGroupName)
            .target(target)
            .actionAfterCompletion(deleteAfterCompletion
                ? ActionAfterCompletion.DELETE
                : ActionAfterCompletion.NONE)
            .startDate(startDate)
            .endDate(endDate)
            .flexibleTimeWindow(flexibleTimeWindow)
            .build();

        return getAsyncClient().createSchedule(request)
            .thenApply(response -> {
                logger.info("Successfully created schedule {} in schedule group {},
The ARN is {} ", name, scheduleGroupName, response.scheduleArn());
                return true;
            })
            .whenComplete((result, ex) -> {
                if (ex != null) {
                    if (ex instanceof ConflictException) {
                        // Handle ConflictException
                        logger.error("A conflict exception occurred while creating
the schedule: {}", ex.getMessage());
                        throw new CompletionException("A conflict exception occurred
while creating the schedule: " + ex.getMessage(), ex);
                    } else {
                        throw new CompletionException("Error creating schedule: " +
ex.getMessage(), ex);
                    }
                }
            });
    }


    /**
     * Deletes the specified schedule group.
     *
     * @param name the name of the schedule group to delete
     * @return a {@link CompletableFuture} that completes when the schedule group
has been deleted
     * @throws CompletionException if an error occurs while deleting the schedule
group
     */
    public CompletableFuture<Void> deleteScheduleGroupAsync(String name) {
```

```
        DeleteScheduleGroupRequest request = DeleteScheduleGroupRequest.builder()
            .name(name)
            .build();

        return getAsyncClient().deleteScheduleGroup(request)
            .thenRun(() -> {
                logger.info("Successfully deleted schedule group {}", name);
            })
            .whenComplete((result, ex) -> {
                if (ex != null) {
                    if (ex instanceof ResourceNotFoundException) {
                        throw new CompletionException("The resource was not found: "
+ ex.getMessage(), ex);
                    } else {
                        throw new CompletionException("Error deleting schedule
group: " + ex.getMessage(), ex);
                    }
                }
            });
    }


    /**
     * Deletes a schedule with the specified name and group name.
     *
     * @param name      the name of the schedule to be deleted
     * @param groupName the group name of the schedule to be deleted
     * @return a {@link CompletableFuture} that, when completed, indicates whether
the schedule was successfully deleted
     * @throws CompletionException if an error occurs while deleting the schedule,
except for the case where the schedule is not found
     */
    public CompletableFuture<Boolean> deleteScheduleAsync(String name, String
groupName) {
        DeleteScheduleRequest request = DeleteScheduleRequest.builder()
            .name(name)
            .groupName(groupName)
            .build();

        CompletableFuture<DeleteScheduleResponse> response =
getAsyncClient().deleteSchedule(request);
        return response.handle((result, ex) -> {
            if (ex != null) {
                if (ex instanceof ResourceNotFoundException) {
```

```
                    throw new CompletionException("Resource not found while deleting
 schedule with ID: " + name, ex);
                } else {
                    throw new CompletionException("Failed to delete schedule.", ex);
                }
            }
            logger.info("Successfully deleted schedule with name {}.", name);
            return true;
        });
    }
}
```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella Documentazione di riferimento delle API AWS SDK for Java 2.x .
  - CreateSchedule
  - CreateScheduleGroup
  - DeleteSchedule
  - DeleteScheduleGroups

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.

Ogni esempio include un collegamento al codice sorgente completo, in cui 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, CreateDatasetconsulta 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, CreateForecastconsulta 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, DeleteDatasetconsulta 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, DeleteForecastconsulta 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, DescribeForecastconsulta 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, ListDatasetGroupsconsulta 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, ListForecastsconsulta 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 nozioni di base sono esempi di codice che mostrano come eseguire le operazioni essenziali all'interno di un servizio.

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.

Ogni esempio include un collegamento al codice sorgente completo, in cui 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, ListJobsconsulta AWS SDK for Java 2.x API Reference.

Argomenti

- Nozioni di base
- Azioni

Nozioni di base

Informazioni di base

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.
 * <p>
 * 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>
 <scriptLocation> <locationUri> <bucketNameSc>\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, s3://<bucket name>/read).
                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 (you can find this file
in resources folder).
                bucketNameSc - The Amazon S3 bucket name used when creating a job
                """;

        if (args.length != 9) {
            System.out.println(usage);
            return;
        }
        Scanner scanner = new Scanner(System.in);
        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("""
            AWS Glue is a fully managed extract, transform, and load (ETL) service
provided by Amazon
            Web Services (AWS). It is designed to simplify the process of building,
running, and maintaining
            ETL pipelines, which are essential for data integration and data
warehousing tasks.

            One of the key features of AWS Glue is its ability to automatically
discover and catalog data
            stored in various sources, such as Amazon S3, Amazon RDS, Amazon
Redshift, and other databases.
            This cataloging process creates a central metadata repository, known as
the AWS Glue Data Catalog,
            which provides a unified view of an organization's data assets. This
metadata can then be used to
            create ETL jobs, which can be scheduled and run on-demand or on a
regular basis.
```

```
            Lets get started.

            """);
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("1. Create a database.");
        try {
            createDatabase(glueClient, dbName, locationUri);
        } catch (GlueException e) {
            if (e.awsErrorDetails().errorMessage().equals("Database already
exists.")) {
                System.out.println("Database " + dbName + " already exists. Skipping
creation.");
            } else {
                System.err.println(e.awsErrorDetails().errorMessage());
                return;
            }
        }

        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("2. Create a crawler.");
        try {
            createGlueCrawler(glueClient, iam, s3Path, cron, dbName, crawlerName);
        } catch (GlueException e) {
            if (e.awsErrorDetails().errorMessage().contains("already exists")) {
                System.out.println("Crawler " + crawlerName + " already exists.
Skipping creation.");
            } else {
                System.err.println(e.awsErrorDetails().errorMessage());
                System.exit(1);
            }
        }
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. Get a crawler.");
        try {
```

```
            getSpecificCrawler(glueClient, crawlerName);
        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            return;
        }
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. Start a crawler.");
        try {
            startSpecificCrawler(glueClient, crawlerName);
        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            return;
        }
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. Get a database.");
        try {
            getSpecificDatabase(glueClient, dbName);
        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            return;
        }
        waitForInputToContinue(scanner);
        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;
        try {
            myTableName = getGlueTables(glueClient, dbName);
        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            return;
        }
        waitForInputToContinue(scanner);
        System.out.println(DASHES);
```

```
        System.out.println(DASHES);
        System.out.println("7. Create a job.");
        try {
            createJob(glueClient, jobName, iam, scriptLocation);
        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            return;
        }
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("8. Start a Job run.");
        try {
            startJob(glueClient, jobName, dbName, myTableName, bucketNameSc);
        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            return;
        }
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("9. List all jobs.");
        try {
            getAllJobs(glueClient);
        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            return;
        }
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("10. Get job runs.");
        try {
            getJobRuns(glueClient, jobName);
        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            return;
        }
        waitForInputToContinue(scanner);
        System.out.println(DASHES);
```

```
        System.out.println(DASHES);
        System.out.println("11. Delete a job.");
        try {
            deleteJob(glueClient, jobName);
        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            return;
        }
        System.out.println("*** Wait 5 MIN for the " + crawlerName + " to stop");
        TimeUnit.MINUTES.sleep(5);
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("12. Delete a database.");
        try {
            deleteDatabase(glueClient, dbName);
        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            return;
        }
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Delete a crawler.");
        try {
            deleteSpecificCrawler(glueClient, crawlerName);
        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            return;
        }
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Successfully completed the AWS Glue Scenario");
        System.out.println(DASHES);
    }


    /**
     * Creates a Glue database with the specified name and location URI.
     *
```

```
    * @param glueClient  The Glue client to use for the database creation.
    * @param dbName      The name of the database to create.
    * @param locationUri The location URI for the database.
    */
   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) {
           throw e;
       }
   }


   /**
    * Creates a new AWS Glue crawler using the AWS Glue Java API.
    *
    * @param glueClient  the AWS Glue client used to interact with the AWS Glue
service
    * @param iam         the IAM role that the crawler will use to access the data
source
    * @param s3Path      the S3 path that the crawler will scan for data
    * @param cron        the cron expression that defines the crawler's schedule
    * @param dbName      the name of the AWS Glue database where the crawler will
store the metadata
    * @param crawlerName the name of the crawler to be created
    */
   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) {
            throw e;
        }
    }

    /**
     * Retrieves a specific crawler from the AWS Glue service and waits for it to be
in the "READY" state.
     *
     * @param glueClient  the AWS Glue client used to interact with the Glue service
     * @param crawlerName the name of the crawler to be retrieved
     */
    public static void getSpecificCrawler(GlueClient glueClient, String crawlerName)
throws InterruptedException {
        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) {
            throw e;
        }
    }

    /**
     * Starts a specific AWS Glue crawler.
     *
     * @param glueClient  the AWS Glue client to use for the crawler operation
     * @param crawlerName the name of the crawler to start
     * @throws GlueException if there is an error starting the crawler
     */
    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) {
            throw e;
        }
    }

    /**
     * Retrieves the specific database from the AWS Glue service.
     *
     * @param glueClient   an instance of the AWS Glue client used to interact with
the service
```

```
    * @param databaseName the name of the database to retrieve
    * @throws GlueException if there is an error retrieving the database from the
AWS Glue service
    */
   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) {
           throw e;
       }
   }


   /**
    * Retrieves the names of the tables in the specified Glue database.
    *
    * @param glueClient the Glue client to use for the operation
    * @param dbName     the name of the Glue database to retrieve the table names
from
    * @return the name of the first table retrieved, or an empty string if no
tables were found
    */
   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) {
            throw e;
        }
        return myTableName;
    }


    /**
     * Starts a job run in AWS Glue.
     *
     * @param glueClient    the AWS Glue client to use for the job run
     * @param jobName       the name of the Glue job to run
     * @param inputDatabase the name of the input database
     * @param inputTable    the name of the input table
     * @param outBucket     the URL of the output S3 bucket
     * @throws GlueException if there is an error starting the job run
     */
    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) {
            throw e;
        }
    }


    /**
     * Creates a new AWS Glue job.
     *
     * @param glueClient     the AWS Glue client to use for the operation
     * @param jobName        the name of the job to create
     * @param iam            the IAM role to associate with the job
     * @param scriptLocation the location of the script to be used by the job
     * @throws GlueException if there is an error creating the job
     */
    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) {
            throw e;
```

```
        }
    }


    /**
     * Retrieves and prints information about all the jobs in the Glue data catalog.
     *
     * @param glueClient the Glue client used to interact with the AWS Glue service
     */
    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) {
            throw e;
        }
    }

    /**
     * Retrieves the job runs for a given Glue job and prints the status of the job
runs.
     *
     * @param glueClient the Glue client used to make API calls
     * @param jobName    the name of the Glue job to retrieve the job runs for
     */
    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 e) {
            throw e;
        } catch (InterruptedException e) {
            throw new RuntimeException(e);
        }
    }


    /**
     * Deletes a Glue job.
     *
     * @param glueClient the Glue client to use for the operation
     * @param jobName    the name of the job to be deleted
```

```
     * @throws GlueException if there is an error deleting the job
     */
    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) {
            throw e;
        }
    }

    /**
     * Deletes a AWS Glue Database.
     *
     * @param glueClient   An instance of the AWS Glue client used to interact with
the AWS Glue service.
     * @param databaseName The name of the database to be deleted.
     * @throws GlueException If an error occurs while deleting the database.
     */
    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) {
            throw e;
        }
    }


    /**
     * Deletes a specific AWS Glue crawler.
     *
     * @param glueClient  the AWS Glue client object
     * @param crawlerName the name of the crawler to be deleted
```

```
     * @throws GlueException if an error occurs during the deletion process
     */
    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) {
            throw e;
        }
    }

    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.");
            }
        }
    }
}
```

- 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

Azioni

`CreateCrawler`

Il seguente esempio di codice mostra come usare`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.

```
    /**
     * Creates a new AWS Glue crawler using the AWS Glue Java API.
     *
     * @param glueClient  the AWS Glue client used to interact with the AWS Glue
service
     * @param iam         the IAM role that the crawler will use to access the data
source
```

```
     * @param s3Path      the S3 path that the crawler will scan for data
     * @param cron        the cron expression that defines the crawler's schedule
     * @param dbName      the name of the AWS Glue database where the crawler will
store the metadata
     * @param crawlerName the name of the crawler to be created
     */
    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) {
            throw e;
        }
    }
```

- Per i dettagli sull'API, CreateCrawlerconsulta AWS SDK for Java 2.x API Reference.

`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.

```
   /**
    * Creates a new AWS Glue job.
    *
    * @param glueClient     the AWS Glue client to use for the operation
    * @param jobName        the name of the job to create
    * @param iam            the IAM role to associate with the job
    * @param scriptLocation the location of the script to be used by the job
    * @throws GlueException if there is an error creating the job
    */
   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) {
            throw e;
        }
    }
```

- Per i dettagli sull'API, CreateJobconsulta AWS SDK for Java 2.x API Reference.

`DeleteCrawler`

Il seguente esempio di codice mostra come utilizzare`DeleteCrawler`.

SDK per Java 2.x

> **Note**
>
> C'è 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 specific AWS Glue crawler.
     *
     * @param glueClient  the AWS Glue client object
     * @param crawlerName the name of the crawler to be deleted
     * @throws GlueException if an error occurs during the deletion process
     */
    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) {
```

```
            throw e;
        }
    }
```

- Per i dettagli sull'API, DeleteCrawlerconsulta AWS SDK for Java 2.x API Reference.

`DeleteDatabase`

Il seguente esempio di codice mostra come utilizzare`DeleteDatabase`.

SDK per Java 2.x

> **Note**
>
> C'è 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 AWS Glue Database.
     *
     * @param glueClient   An instance of the AWS Glue client used to interact with
the AWS Glue service.
     * @param databaseName The name of the database to be deleted.
     * @throws GlueException If an error occurs while deleting the database.
     */
    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) {
            throw e;
        }
    }
```

- Per i dettagli sull'API, DeleteDatabaseconsulta AWS SDK for Java 2.x API Reference.

`DeleteJob`

Il seguente esempio di codice mostra come utilizzare`DeleteJob`.

SDK per Java 2.x

> **Note**
>
> C'è 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 Glue job.
 *
 * @param glueClient the Glue client to use for the operation
 * @param jobName    the name of the job to be deleted
 * @throws GlueException if there is an error deleting the job
 */
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) {
        throw e;
    }
}
```

- Per i dettagli sull'API, DeleteJobconsulta 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.

```
   /**
    * Retrieves a specific crawler from the AWS Glue service and waits for it to be
in the "READY" state.
    *
    * @param glueClient  the AWS Glue client used to interact with the Glue service
    * @param crawlerName the name of the crawler to be retrieved
    */
   public static void getSpecificCrawler(GlueClient glueClient, String crawlerName)
throws InterruptedException {
       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) {
           throw e;
       }
   }
```

- Per i dettagli sull'API, GetCrawlerconsulta 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.

```
   /**
    * Retrieves the specific database from the AWS Glue service.
    *
    * @param glueClient   an instance of the AWS Glue client used to interact with
the service
    * @param databaseName the name of the database to retrieve
    * @throws GlueException if there is an error retrieving the database from the
AWS Glue service
    */
   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) {
            throw e;
        }
    }
```

- Per i dettagli sull'API, GetDatabaseconsulta AWS SDK for Java 2.x API Reference.

`GetJobRuns`

Il seguente esempio di codice mostra come utilizzare`GetJobRuns`.

SDK per Java 2.x

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
    /**
     * Retrieves the job runs for a given Glue job and prints the status of the job
runs.
     *
     * @param glueClient the Glue client used to make API calls
     * @param jobName    the name of the Glue job to retrieve the job runs for
     */
    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 e) {
            throw e;
        } catch (InterruptedException e) {
            throw new RuntimeException(e);
        }
    }
```

- Per i dettagli sull'API, GetJobRunsconsulta 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.

```
   /**
    * Retrieves the names of the tables in the specified Glue database.
    *
    * @param glueClient the Glue client to use for the operation
    * @param dbName     the name of the Glue database to retrieve the table names
from
    * @return the name of the first table retrieved, or an empty string if no
tables were found
    */
   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) {
           throw e;
       }
       return myTableName;
   }
```

- Per i dettagli sull'API, GetTablesconsulta 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.

```
   /**
    * Starts a specific AWS Glue crawler.
    *
    * @param glueClient  the AWS Glue client to use for the crawler operation
    * @param crawlerName the name of the crawler to start
    * @throws GlueException if there is an error starting the crawler
    */
   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) {
           throw e;
       }
   }
```

- Per i dettagli sull'API, StartCrawlerconsulta AWS SDK for Java 2.x API Reference.

`StartJobRun`

Il seguente esempio di codice mostra come utilizzare`StartJobRun`.

SDK per Java 2.x

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
   /**
    * Starts a job run in AWS Glue.
    *
    * @param glueClient    the AWS Glue client to use for the job run
    * @param jobName       the name of the Glue job to run
    * @param inputDatabase the name of the input database
    * @param inputTable    the name of the input table
    * @param outBucket     the URL of the output S3 bucket
    * @throws GlueException if there is an error starting the job run
    */
   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) {
            throw e;
        }
    }
```

- Per i dettagli sull'API, StartJobRunconsulta AWS SDK for Java 2.x API Reference.

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.

Gli scenari sono esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio o combinate con altri Servizi AWS.

Ogni esempio include un collegamento al codice sorgente completo, in cui 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

```
/**
 * Copy an AWS HealthImaging image set.
 *
 * @param medicalImagingClient  - The AWS HealthImaging client object.
```

```
    * @param datastoreId           - The datastore ID.
    * @param imageSetId            - The image set ID.
    * @param latestVersionId       - The version ID.
    * @param destinationImageSetId - The optional destination image set ID, ignored
if null.
    * @param destinationVersionId  - The optional destination version ID, ignored
if null.
    * @param force                 - The force flag.
    * @param subsets               - The optional subsets to copy, ignored if null.
    * @return                      - The image set ID of the copy.
    * @throws MedicalImagingException - Base exception for all service exceptions
thrown by AWS HealthImaging.
    */
   public static String copyMedicalImageSet(MedicalImagingClient
medicalImagingClient,
                                            String datastoreId,
                                            String imageSetId,
                                            String latestVersionId,
                                            String destinationImageSetId,
                                            String destinationVersionId,
                                            boolean force,
                                            Vector<String> subsets) {

       try {
           CopySourceImageSetInformation.Builder copySourceImageSetInformation =
CopySourceImageSetInformation.builder()
                   .latestVersionId(latestVersionId);

           // Optionally copy a subset of image instances.
           if (subsets != null) {
               String subsetInstanceToCopy = getCopiableAttributesJSON(imageSetId,
subsets);
               copySourceImageSetInformation.dicomCopies(MetadataCopies.builder()
                       .copiableAttributes(subsetInstanceToCopy)
                       .build());
           }

           CopyImageSetInformation.Builder copyImageSetBuilder =
CopyImageSetInformation.builder()
                   .sourceImageSet(copySourceImageSetInformation.build());

           // Optionally designate a destination image set.
           if (destinationImageSetId != null) {
```

```
                copyImageSetBuilder =
copyImageSetBuilder.destinationImageSet(CopyDestinationImageSet.builder()
                        .imageSetId(destinationImageSetId)
                        .latestVersionId(destinationVersionId)
                        .build());
            }

            CopyImageSetRequest copyImageSetRequest = CopyImageSetRequest.builder()
                    .datastoreId(datastoreId)
                    .sourceImageSetId(imageSetId)
                    .copyImageSetInformation(copyImageSetBuilder.build())
                    .force(force)
                    .build();

            CopyImageSetResponse response =
medicalImagingClient.copyImageSet(copyImageSetRequest);

            return response.destinationImageSetProperties().imageSetId();
        } catch (MedicalImagingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            throw e;
        }
    }
```

Funzione di utilità per creare attributi copiabili.

```
    /**
     * Create a JSON string of copiable image instances.
     *
     * @param imageSetId - The image set ID.
     * @param subsets    - The subsets to copy.
     * @return A JSON string of copiable image instances.
     */
    private static String getCopiableAttributesJSON(String imageSetId,
Vector<String> subsets) {
        StringBuilder subsetInstanceToCopy = new StringBuilder(
                """
                        {
                          "SchemaVersion": 1.1,
                          "Study": {
                            "Series": {
```

```
                                "
                                 """
        );

        subsetInstanceToCopy.append(imageSetId);

        subsetInstanceToCopy.append(
                """
                                ": {
                                "Instances": {
                        """
        );

        for (String subset : subsets) {
            subsetInstanceToCopy.append('"' + subset + "\": {},");
        }
        subsetInstanceToCopy.deleteCharAt(subsetInstanceToCopy.length() - 1);
        subsetInstanceToCopy.append("""
                          }
                        }
                      }
                    }
                  }
                """);
        return subsetInstanceToCopy.toString();
    }
```

- Per i dettagli sull'API, vedere CopyImageSetin 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 CreateDatastoresezione 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 DeleteDatastoresezione 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 DeleteImageSetsezione 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 Get DICOMImport Job 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.

`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 GetDatastoresezione 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 GetImageFramesezione 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 GetImageSetsezione 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 GetImageSetMetadatasezione 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 List DICOMImport Jobs 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.

`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 ListDatastoressezione 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 DICOMStudy data e DICOMStudy ora.

```
        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 DICOMSeries instanceUID 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, consulta la sezione API Reference. SearchImageSetsAWS 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 Start DICOMImport Job 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 UntagResourcesezione 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

```
    /**
     * Update the metadata of an AWS HealthImaging image set.
     *
     * @param medicalImagingClient - The AWS HealthImaging client object.
     * @param datastoreId          - The datastore ID.
     * @param imageSetId           - The image set ID.
     * @param versionId            - The version ID.
```

```
    * @param metadataUpdates      - A MetadataUpdates object containing the
updates.
    * @param force                - The force flag.
    * @throws MedicalImagingException - Base exception for all service exceptions
thrown by AWS HealthImaging.
    */
   public static void updateMedicalImageSetMetadata(MedicalImagingClient
medicalImagingClient,
                                                    String datastoreId,
                                                    String imageSetId,
                                                    String versionId,
                                                    MetadataUpdates
metadataUpdates,
                                                    boolean force) {
       try {
           UpdateImageSetMetadataRequest updateImageSetMetadataRequest =
UpdateImageSetMetadataRequest
                   .builder()
                   .datastoreId(datastoreId)
                   .imageSetId(imageSetId)
                   .latestVersionId(versionId)
                   .updateImageSetMetadataUpdates(metadataUpdates)
                   .force(force)
                   .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());
           throw e;
       }
   }
```

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()
                              .updatableAttributes(SdkBytes.fromByteBuffer(
                                      ByteBuffer.wrap(insertAttributes
                                              .getBytes(StandardCharsets.UTF_8))))
                              .build())
                      .build();

              updateMedicalImageSetMetadata(medicalImagingClient, datastoreId,
imagesetId,
                      versionid, metadataInsertUpdates, force);
```

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, force);
```

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, force);
```

Caso d'uso #4: ripristina una versione precedente.

```
                // In this case, revert to previous version.
                String revertVersionId =
Integer.toString(Integer.parseInt(versionid) - 1);
                MetadataUpdates metadataRemoveUpdates = MetadataUpdates.builder()
                        .revertToVersionId(revertVersionId)
                        .build();
                updateMedicalImageSetMetadata(medicalImagingClient, datastoreId,
imagesetId,
                        versionid, metadataRemoveUpdates, force);
```

- Per i dettagli sull'API, consulta la sezione UpdateImageSetMetadata AWS SDK for Java 2.xAPI 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 nozioni di base sono esempi di codice che mostrano come eseguire le operazioni essenziali all'interno di un servizio.

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.

Gli scenari sono esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio o combinate con altri Servizi AWS.

Ogni esempio include un collegamento al codice sorgente completo, in cui 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, ListPoliciesconsulta AWS SDK for Java 2.x API Reference.

Argomenti

- Nozioni di base
- Azioni
- Scenari

Nozioni di base

Informazioni di base

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

> **Note**
>
> C'è 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

Azioni

`AttachRolePolicy`

Il seguente esempio di codice mostra come usare`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, AttachRolePolicyconsulta 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, CreateAccessKeyconsulta 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, CreateAccountAliasconsulta 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, CreatePolicyconsulta 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, CreateRoleconsulta 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, CreateUserconsulta 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, DeleteAccessKeyconsulta 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, DeleteAccountAliasconsulta 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, DeletePolicyconsulta 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, DeleteUserconsulta 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, DetachRolePolicyconsulta 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, ListAccessKeysconsulta 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, ListAccountAliasesconsulta 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, ListUsersconsulta 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, UpdateAccessKeyconsulta 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, UpdateUserconsulta 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 Amazon EC2 Auto Scaling per creare istanze Amazon Elastic Compute Cloud (Amazon EC2) basate su un modello di avvio e per mantenere il numero di istanze in un intervallo specificato.
- Gestisci e distribuisci 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 EC2 istanza 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 di integrità aggiornando AWS Systems Manager i parametri.

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.

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 wasSuccessul = loadBalancer.verifyLoadBalancerEndpoint(elbDnsName);
        if (!wasSuccessul) {
            System.out.println("Couldn't connect to the load balancer, verifying
that the port is open...");
            CloseableHttpClient httpClient = HttpClients.createDefault();

            // Create an HTTP GET request to "http://checkip.amazonaws.com"
            HttpGet httpGet = new HttpGet("http://checkip.amazonaws.com");
            try {
                // Execute the request and get the response
                HttpResponse response = httpClient.execute(httpGet);

                // Read the response content.
                String ipAddress =
IOUtils.toString(response.getEntity().getContent(), StandardCharsets.UTF_8).trim();

                // Print the public IP address.
                System.out.println("Public IP Address: " + ipAddress);
```

```
                GroupInfo groupInfo = autoScaler.verifyInboundPort(vpcId, port,
ipAddress);
                if (!groupInfo.isPortOpen()) {
                    System.out.println("""
                            For this example to work, the default security group for
your default VPC must
                            allow access from this computer. You can either add it
automatically from this
                            example or add it yourself using the AWS Management
Console.
                            """);

                    System.out.println(
                            "Do you want to add a rule to security group " +
groupInfo.getGroupName() + " to allow");
                    System.out.println("inbound traffic on port " + port + " from
your computer's IP address (y/n) ");
                    String ans = in.nextLine();
                    if ("y".equalsIgnoreCase(ans)) {
                        autoScaler.openInboundPort(groupInfo.getGroupName(),
String.valueOf(port), ipAddress);
                        System.out.println("Security group rule added.");
                    } else {
                        System.out.println("No security group rule added.");
                    }
                }

            } catch (AutoScalingException e) {
                e.printStackTrace();
            }
        } else if (wasSuccessul) {
            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 azioni di Auto Scaling e EC2 Amazon.

```
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.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)
                    .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

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 di più su. 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:::amzn-s3-
demo-bucket/*")
                                                .addResource("arn:aws:s3:::amzn-s3-
demo-bucket")
                                                .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:::amzn-s3-
demo-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 nozioni di base sono esempi di codice che mostrano come eseguire le operazioni essenziali all'interno di un servizio.

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.

Ogni esempio include un collegamento al codice sorgente completo, in cui 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 software.amazon.awssdk.services.iot.paginators.ListThingsIterable;

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) {
        iotClient.listThingsPaginator(ListThingsRequest.builder()
                .maxResults(10)
                .build())
            .stream()
            .flatMap(response -> response.things().stream())
            .forEach(attribute -> {
                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

- Nozioni di base
- Azioni

Nozioni di base

Informazioni di base

L'esempio di codice seguente mostra come:

- Crea un AWS IoT oggetto.
- Genera un certificato del dispositivo.
- Aggiorna un AWS IoT oggetto con attributi.
- Restituisce un endpoint univoco.
- Elenca i tuoi AWS IoT certificati.
- Crea un' AWS IoT ombra.
- Scrivi informazioni sullo stato.
- Crea una regola.
- Elenca le tue regole.
- Cerca elementi utilizzando il nome dell'oggetto.
- Eliminare un AWS IoT 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.

Esegui uno scenario interattivo che dimostri le AWS IoT funzionalità.

```
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
```

```
 *
 * 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 amd delete the certificate.
 * 12. Delete Thing.
 */
public class IotScenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    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);
        }

        IotActions iotActions = new IotActions();
        String thingName;
        String ruleName;
        String roleARN = args[0];
        String snsAction = args[1];
        Scanner scanner = new Scanner(System.in);

        System.out.println(DASHES);
```

```
        System.out.println("Welcome to the AWS IoT basics scenario.");
        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.

            Let's get started...

            """);
        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();
        iotActions.createIoTThing(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 = iotActions.createCertificate();
```

```
            System.out.println("Attach the certificate to the AWS IoT Thing.");
            iotActions.attachCertificateToThing(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.
            """);
        waitForInputToContinue(scanner);
        iotActions.updateShadowThing(thingName);
        waitForInputToContinue(scanner);
        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.
           """);
        waitForInputToContinue(scanner);
        String endpointUrl = iotActions.describeEndpoint();
        System.out.println("The endpoint is "+endpointUrl);
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. List your AWS IoT certificates");
        waitForInputToContinue(scanner);
        if (certificateArn.length() > 0) {
            iotActions.listCertificates();
        } else {
            System.out.println("You did not create a certificates. Skipping this
step.");
        }
        waitForInputToContinue(scanner);
        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.
           """);
        waitForInputToContinue(scanner);
        iotActions.updateShadowThing(thingName);
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Write out the state information, in JSON format.");
        waitForInputToContinue(scanner);
        iotActions.getPayload(thingName);
        waitForInputToContinue(scanner);
        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();
        iotActions.createIoTRule(roleARN, ruleName, snsAction);
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("9. List your rules.");
        waitForInputToContinue(scanner);
        iotActions.listIoTRules();
        waitForInputToContinue(scanner);
        System.out.println(DASHES);
```

```
        System.out.println(DASHES);
        System.out.println("10. Search things using the Thing name.");
        waitForInputToContinue(scanner);
        String queryString = "thingName:"+thingName ;
        iotActions.searchThings(queryString);
        waitForInputToContinue(scanner);
        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.");
                waitForInputToContinue(scanner);
                iotActions.detachThingPrincipal(thingName, certificateArn);
                iotActions.deleteCertificate(certificateArn);
                waitForInputToContinue(scanner);
            } 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")) {
            iotActions.deleteIoTThing(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);
```

```
    }


    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.");
            }
        }
    }
}
```

Una classe wrapper per i metodi SDK. AWS IoT

```
import
 software.amazon.awssdk.auth.credentials.EnvironmentVariableCredentialsProvider;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.core.client.config.ClientOverrideConfiguration;
import software.amazon.awssdk.core.retry.RetryPolicy;
import software.amazon.awssdk.http.async.SdkAsyncHttpClient;
import software.amazon.awssdk.http.nio.netty.NettyNioAsyncHttpClient;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iot.IotAsyncClient;
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.Certificate;
import software.amazon.awssdk.services.iot.model.CreateKeysAndCertificateResponse;
import software.amazon.awssdk.services.iot.model.CreateThingRequest;
import software.amazon.awssdk.services.iot.model.CreateThingResponse;
import software.amazon.awssdk.services.iot.model.CreateTopicRuleRequest;
import software.amazon.awssdk.services.iot.model.CreateTopicRuleResponse;
import software.amazon.awssdk.services.iot.model.DeleteCertificateRequest;
```

```
import software.amazon.awssdk.services.iot.model.DeleteCertificateResponse;
import software.amazon.awssdk.services.iot.model.DeleteThingRequest;
import software.amazon.awssdk.services.iot.model.DeleteThingResponse;
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.DetachThingPrincipalResponse;
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.TopicRuleListItem;
import software.amazon.awssdk.services.iot.model.SnsAction;
import software.amazon.awssdk.services.iot.model.TopicRulePayload;
import software.amazon.awssdk.services.iotdataplane.IotDataPlaneAsyncClient;
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 software.amazon.awssdk.services.iotdataplane.model.UpdateThingShadowResponse;
import java.nio.charset.StandardCharsets;
import java.time.Duration;
import java.util.List;
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.CompletionException;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

public class IotActions {

    private static IotAsyncClient iotAsyncClient;

    private static IotDataPlaneAsyncClient iotAsyncDataPlaneClient;

    private static final String TOPIC = "your-iot-topic";

    private static IotDataPlaneAsyncClient getAsyncDataPlaneClient() {
        SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
            .maxConcurrency(100)
            .connectionTimeout(Duration.ofSeconds(60))
            .readTimeout(Duration.ofSeconds(60))
```

```
            .writeTimeout(Duration.ofSeconds(60))
            .build();

        ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
            .apiCallTimeout(Duration.ofMinutes(2))
            .apiCallAttemptTimeout(Duration.ofSeconds(90))
            .retryPolicy(RetryPolicy.builder()
                .numRetries(3)
                .build())
            .build();

        if (iotAsyncDataPlaneClient == null) {
            iotAsyncDataPlaneClient = IotDataPlaneAsyncClient.builder()
                .region(Region.US_EAST_1)
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return iotAsyncDataPlaneClient;
    }


    private static IotAsyncClient getAsyncClient() {
        SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
            .maxConcurrency(100)
            .connectionTimeout(Duration.ofSeconds(60))
            .readTimeout(Duration.ofSeconds(60))
            .writeTimeout(Duration.ofSeconds(60))
            .build();

        ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
            .apiCallTimeout(Duration.ofMinutes(2))
            .apiCallAttemptTimeout(Duration.ofSeconds(90))
            .retryPolicy(RetryPolicy.builder()
                .numRetries(3)
                .build())
            .build();

        if (iotAsyncClient == null) {
            iotAsyncClient = IotAsyncClient.builder()
                .region(Region.US_EAST_1)
                .httpClient(httpClient)
```

```
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return iotAsyncClient;
    }

    /**
     * Creates an IoT certificate asynchronously.
     *
     * @return The ARN of the created certificate.
     * <p>
     * This method initiates an asynchronous request to create an IoT certificate.
     * If the request is successful, it prints the certificate details and returns
the certificate ARN.
     * If an exception occurs, it prints the error message.
     */
    public String createCertificate() {
        CompletableFuture<CreateKeysAndCertificateResponse> future =
getAsyncClient().createKeysAndCertificate();
        final String[] certificateArn = {null};
        future.whenComplete((response, ex) -> {
            if (response != null) {
                String certificatePem = response.certificatePem();
                certificateArn[0] = response.certificateArn();

                // Print the details.
                System.out.println("\nCertificate:");
                System.out.println(certificatePem);
                System.out.println("\nCertificate ARN:");
                System.out.println(certificateArn[0]);

            } else {
                Throwable cause = (ex instanceof CompletionException) ?
ex.getCause() : ex;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else {
                    System.err.println("Unexpected error: " + cause.getMessage());
                }
            }
        });

        future.join();
```

```
        return certificateArn[0];
    }

    /**
     * Creates an IoT Thing with the specified name asynchronously.
     *
     * @param thingName The name of the IoT Thing to create.
     *
     * This method initiates an asynchronous request to create an IoT Thing with the
specified name.
     * If the request is successful, it prints the name of the thing and its ARN
value.
     * If an exception occurs, it prints the error message.
     */
    public void createIoTThing(String thingName) {
        CreateThingRequest createThingRequest = CreateThingRequest.builder()
            .thingName(thingName)
            .build();

        CompletableFuture<CreateThingResponse> future =
getAsyncClient().createThing(createThingRequest);
        future.whenComplete((createThingResponse, ex) -> {
            if (createThingResponse != null) {
                System.out.println(thingName + " was successfully created. The ARN
value is " + createThingResponse.thingArn());
            } else {
                Throwable cause = ex.getCause();
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else {
                    System.err.println("Unexpected error: " + cause.getMessage());
                }
            }
        });

        future.join();
    }

    /**
     * Attaches a certificate to an IoT Thing asynchronously.
     *
     * @param thingName The name of the IoT Thing.
     * @param certificateArn The ARN of the certificate to attach.
```

```
    *
    * This method initiates an asynchronous request to attach a certificate to an
IoT Thing.
    * If the request is successful, it prints a confirmation message and additional
information about the Thing.
    * If an exception occurs, it prints the error message.
    */
   public void attachCertificateToThing(String thingName, String certificateArn) {
       AttachThingPrincipalRequest principalRequest =
AttachThingPrincipalRequest.builder()
           .thingName(thingName)
           .principal(certificateArn)
           .build();

       CompletableFuture<AttachThingPrincipalResponse> future =
getAsyncClient().attachThingPrincipal(principalRequest);
       future.whenComplete((attachResponse, ex) -> {
           if (attachResponse != null &&
attachResponse.sdkHttpResponse().isSuccessful()) {
               System.out.println("Certificate attached to Thing successfully.");

               // Print additional information about the Thing.
               describeThing(thingName);
           } else {
               Throwable cause = ex != null ? ex.getCause() : null;
               if (cause instanceof IotException) {
                   System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
               } else if (cause != null) {
                   System.err.println("Unexpected error: " + cause.getMessage());
               } else {
                   System.err.println("Failed to attach certificate to Thing. HTTP
Status Code: " +
                           attachResponse.sdkHttpResponse().statusCode());
               }
           }
       });

       future.join();
   }

   /**
    * Describes an IoT Thing asynchronously.
    *
```

```
     * @param thingName The name of the IoT Thing.
     *
     * This method initiates an asynchronous request to describe an IoT Thing.
     * If the request is successful, it prints the Thing details.
     * If an exception occurs, it prints the error message.
     */
    private void describeThing(String thingName) {
        DescribeThingRequest thingRequest = DescribeThingRequest.builder()
            .thingName(thingName)
            .build();

        CompletableFuture<DescribeThingResponse> future =
getAsyncClient().describeThing(thingRequest);
        future.whenComplete((describeResponse, ex) -> {
            if (describeResponse != null) {
                System.out.println("Thing Details:");
                System.out.println("Thing Name: " + describeResponse.thingName());
                System.out.println("Thing ARN: " + describeResponse.thingArn());
            } else {
                Throwable cause = ex != null ? ex.getCause() : null;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else if (cause != null) {
                    System.err.println("Unexpected error: " + cause.getMessage());
                } else {
                    System.err.println("Failed to describe Thing.");
                }
            }
        });

        future.join();
    }

    /**
     * Updates the shadow of an IoT Thing asynchronously.
     *
     * @param thingName The name of the IoT Thing.
     *
     * This method initiates an asynchronous request to update the shadow of an IoT
Thing.
     * If the request is successful, it prints a confirmation message.
     * If an exception occurs, it prints the error message.
     */
```

```
   public void updateShadowThing(String thingName) {
       // 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();

       CompletableFuture<UpdateThingShadowResponse> future =
getAsyncDataPlaneClient().updateThingShadow(updateThingShadowRequest);
       future.whenComplete((updateResponse, ex) -> {
           if (updateResponse != null) {
               System.out.println("Thing Shadow updated successfully.");
           } else {
               Throwable cause = ex != null ? ex.getCause() : null;
               if (cause instanceof IotException) {
                   System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
               } else if (cause != null) {
                   System.err.println("Unexpected error: " + cause.getMessage());
               } else {
                   System.err.println("Failed to update Thing Shadow.");
               }
           }
       });

       future.join();
   }

   /**
    * Describes the endpoint of the IoT service asynchronously.
    *
    * @return A CompletableFuture containing the full endpoint URL.
    *
    * This method initiates an asynchronous request to describe the endpoint of the
IoT service.
    * If the request is successful, it prints and returns the full endpoint URL.
    * If an exception occurs, it prints the error message.
    */
   public String describeEndpoint() {
```

```
        CompletableFuture<DescribeEndpointResponse> future =
 getAsyncClient().describeEndpoint(DescribeEndpointRequest.builder().endpointType("iot:Data-
ATS").build());
        final String[] result = {null};

        future.whenComplete((endpointResponse, ex) -> {
            if (endpointResponse != null) {
                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);
                result[0] = fullEndpoint;
            } else {
                Throwable cause = (ex instanceof CompletionException) ?
 ex.getCause() : ex;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
 cause).awsErrorDetails().errorMessage());
                } else {
                    System.err.println("Unexpected error: " + cause.getMessage());
                }
            }
        });

        future.join();
        return result[0];
    }

    /**
     * Extracts a specific value from the endpoint URL.
     *
     * @param input The endpoint URL to process.
     * @return The extracted value from the endpoint URL.
     */
    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 "" ;
    }

    /**
     * Lists all certificates asynchronously.
     *
     * This method initiates an asynchronous request to list all certificates.
     * If the request is successful, it prints the certificate IDs and ARNs.
     * If an exception occurs, it prints the error message.
     */
    public void listCertificates() {
        CompletableFuture<ListCertificatesResponse> future =
getAsyncClient().listCertificates();
        future.whenComplete((response, ex) -> {
            if (response != null) {
                List<Certificate> certList = response.certificates();
                for (Certificate cert : certList) {
                    System.out.println("Cert id: " + cert.certificateId());
                    System.out.println("Cert Arn: " + cert.certificateArn());
                }
            } else {
                Throwable cause = ex != null ? ex.getCause() : null;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else if (cause != null) {
                    System.err.println("Unexpected error: " + cause.getMessage());
                } else {
                    System.err.println("Failed to list certificates.");
                }
            }
        });

        future.join();
    }
```

```
   /**
    * Retrieves the payload of a Thing's shadow asynchronously.
    *
    * @param thingName The name of the IoT Thing.
    *
    * This method initiates an asynchronous request to get the payload of a Thing's
shadow.
    * If the request is successful, it prints the shadow data.
    * If an exception occurs, it prints the error message.
    */
   public void getPayload(String thingName) {
       GetThingShadowRequest getThingShadowRequest =
GetThingShadowRequest.builder()
           .thingName(thingName)
           .build();

       CompletableFuture<GetThingShadowResponse> future =
getAsyncDataPlaneClient().getThingShadow(getThingShadowRequest);
       future.whenComplete((getThingShadowResponse, ex) -> {
           if (getThingShadowResponse != null) {
               // Extracting payload from response.
               SdkBytes payload = getThingShadowResponse.payload();
               String payloadString = payload.asUtf8String();
               System.out.println("Received Shadow Data: " + payloadString);
           } else {
               Throwable cause = ex != null ? ex.getCause() : null;
               if (cause instanceof IotException) {
                   System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
               } else if (cause != null) {
                   System.err.println("Unexpected error: " + cause.getMessage());
               } else {
                   System.err.println("Failed to get Thing Shadow payload.");
               }
           }
       });

       future.join();
   }

   /**
    * Creates an IoT rule asynchronously.
    *
```

```
    * @param roleARN The ARN of the IAM role that grants access to the rule's
actions.
    * @param ruleName The name of the IoT rule.
    * @param action The ARN of the action to perform when the rule is triggered.
    *
    * This method initiates an asynchronous request to create an IoT rule.
    * If the request is successful, it prints a confirmation message.
    * If an exception occurs, it prints the error message.
    */
   public void createIoTRule(String roleARN, String ruleName, String action) {
       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();

       CompletableFuture<CreateTopicRuleResponse> future =
getAsyncClient().createTopicRule(topicRuleRequest);
       future.whenComplete((response, ex) -> {
           if (response != null) {
               System.out.println("IoT Rule created successfully.");
           } else {
               Throwable cause = ex != null ? ex.getCause() : null;
               if (cause instanceof IotException) {
                   System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
               } else if (cause != null) {
```

```
                    System.err.println("Unexpected error: " + cause.getMessage());
                } else {
                    System.err.println("Failed to create IoT Rule.");
                }
            }
        });

        future.join();
    }

    /**
     * Lists IoT rules asynchronously.
     *
     * This method initiates an asynchronous request to list IoT rules.
     * If the request is successful, it prints the names and ARNs of the rules.
     * If an exception occurs, it prints the error message.
     */
    public void listIoTRules() {
        ListTopicRulesRequest listTopicRulesRequest =
ListTopicRulesRequest.builder().build();
        CompletableFuture<ListTopicRulesResponse> future =
getAsyncClient().listTopicRules(listTopicRulesRequest);
        future.whenComplete((listTopicRulesResponse, ex) -> {
            if (listTopicRulesResponse != null) {
                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("--------------");
                }
            } else {
                Throwable cause = ex != null ? ex.getCause() : null;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else if (cause != null) {
                    System.err.println("Unexpected error: " + cause.getMessage());
                } else {
                    System.err.println("Failed to list IoT Rules.");
                }
            }
        });
```

```
        future.join();
    }

    /**
     * Searches for IoT Things asynchronously based on a query string.
     *
     * @param queryString The query string to search for Things.
     *
     * This method initiates an asynchronous request to search for IoT Things.
     * If the request is successful and Things are found, it prints their IDs.
     * If no Things are found, it prints a message indicating so.
     * If an exception occurs, it prints the error message.
     */
    public void searchThings(String queryString) {
        SearchIndexRequest searchIndexRequest = SearchIndexRequest.builder()
            .queryString(queryString)
            .build();

        CompletableFuture<SearchIndexResponse> future =
getAsyncClient().searchIndex(searchIndexRequest);
        future.whenComplete((searchIndexResponse, ex) -> {
            if (searchIndexResponse != null) {
                // 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()));
                }
            } else {
                Throwable cause = ex != null ? ex.getCause() : null;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else if (cause != null) {
                    System.err.println("Unexpected error: " + cause.getMessage());
                } else {
                    System.err.println("Failed to search for IoT Things.");
                }
            }
        });

        future.join();
    }
```

```
   /**
    * Detaches a principal (certificate) from an IoT Thing asynchronously.
    *
    * @param thingName The name of the IoT Thing.
    * @param certificateArn The ARN of the certificate to detach.
    *
    * This method initiates an asynchronous request to detach a certificate from an
IoT Thing.
    * If the detachment is successful, it prints a confirmation message.
    * If an exception occurs, it prints the error message.
    */
   public void detachThingPrincipal(String thingName, String certificateArn) {
       DetachThingPrincipalRequest thingPrincipalRequest =
DetachThingPrincipalRequest.builder()
           .principal(certificateArn)
           .thingName(thingName)
           .build();

       CompletableFuture<DetachThingPrincipalResponse> future =
getAsyncClient().detachThingPrincipal(thingPrincipalRequest);
       future.whenComplete((voidResult, ex) -> {
           if (ex == null) {
               System.out.println(certificateArn + " was successfully removed from
" + thingName);
           } else {
               Throwable cause = ex.getCause();
               if (cause instanceof IotException) {
                   System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
               } else {
                   System.err.println("Unexpected error: " + ex.getMessage());
               }
           }
       });

       future.join();
   }

   /**
    * Deletes a certificate asynchronously.
    *
    * @param certificateArn The ARN of the certificate to delete.
    *
```

```
    * This method initiates an asynchronous request to delete a certificate.
    * If the deletion is successful, it prints a confirmation message.
    * If an exception occurs, it prints the error message.
    */
   public void deleteCertificate(String certificateArn) {
       DeleteCertificateRequest certificateProviderRequest =
DeleteCertificateRequest.builder()
           .certificateId(extractCertificateId(certificateArn))
           .build();

       CompletableFuture<DeleteCertificateResponse> future =
getAsyncClient().deleteCertificate(certificateProviderRequest);
       future.whenComplete((voidResult, ex) -> {
           if (ex == null) {
               System.out.println(certificateArn + " was successfully deleted.");
           } else {
               Throwable cause = ex.getCause();
               if (cause instanceof IotException) {
                   System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
               } else {
                   System.err.println("Unexpected error: " + ex.getMessage());
               }
           }
       });

       future.join();
   }

   /**
    * Deletes an IoT Thing asynchronously.
    *
    * @param thingName The name of the IoT Thing to delete.
    *
    * This method initiates an asynchronous request to delete an IoT Thing.
    * If the deletion is successful, it prints a confirmation message.
    * If an exception occurs, it prints the error message.
    */
   public void deleteIoTThing(String thingName) {
       DeleteThingRequest deleteThingRequest = DeleteThingRequest.builder()
           .thingName(thingName)
           .build();
```

```
        CompletableFuture<DeleteThingResponse> future =
 getAsyncClient().deleteThing(deleteThingRequest);
        future.whenComplete((voidResult, ex) -> {
            if (ex == null) {
                System.out.println("Deleted Thing " + thingName);
            } else {
                Throwable cause = ex.getCause();
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
 cause).awsErrorDetails().errorMessage());
                } else {
                    System.err.println("Unexpected error: " + ex.getMessage());
                }
            }
        });

        future.join();
    }

    // Get the cert Id  from the Cert ARN value.
    private 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);
    }
}
```

Azioni

`AttachThingPrincipal`

Il seguente esempio di codice mostra come usare. `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.

```
   /**
    * Attaches a certificate to an IoT Thing asynchronously.
    *
    * @param thingName The name of the IoT Thing.
    * @param certificateArn The ARN of the certificate to attach.
    *
    * This method initiates an asynchronous request to attach a certificate to an
IoT Thing.
    * If the request is successful, it prints a confirmation message and additional
information about the Thing.
    * If an exception occurs, it prints the error message.
    */
   public void attachCertificateToThing(String thingName, String certificateArn) {
       AttachThingPrincipalRequest principalRequest =
AttachThingPrincipalRequest.builder()
           .thingName(thingName)
           .principal(certificateArn)
           .build();

       CompletableFuture<AttachThingPrincipalResponse> future =
getAsyncClient().attachThingPrincipal(principalRequest);
       future.whenComplete((attachResponse, ex) -> {
           if (attachResponse != null &&
attachResponse.sdkHttpResponse().isSuccessful()) {
               System.out.println("Certificate attached to Thing successfully.");

               // Print additional information about the Thing.
               describeThing(thingName);
           } else {
               Throwable cause = ex != null ? ex.getCause() : null;
               if (cause instanceof IotException) {
                   System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
               } else if (cause != null) {
                   System.err.println("Unexpected error: " + cause.getMessage());
               } else {
                   System.err.println("Failed to attach certificate to Thing. HTTP
Status Code: " +
                           attachResponse.sdkHttpResponse().statusCode());
               }
           }
       });
```

```
        future.join();
    }
```

- Per i dettagli sull'API, AttachThingPrincipalconsulta 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.

```
    /**
     * Creates an IoT certificate asynchronously.
     *
     * @return The ARN of the created certificate.
     * <p>
     * This method initiates an asynchronous request to create an IoT certificate.
     * If the request is successful, it prints the certificate details and returns
the certificate ARN.
     * If an exception occurs, it prints the error message.
     */
    public String createCertificate() {
        CompletableFuture<CreateKeysAndCertificateResponse> future =
getAsyncClient().createKeysAndCertificate();
        final String[] certificateArn = {null};
        future.whenComplete((response, ex) -> {
            if (response != null) {
                String certificatePem = response.certificatePem();
                certificateArn[0] = response.certificateArn();

                // Print the details.
                System.out.println("\nCertificate:");
                System.out.println(certificatePem);
                System.out.println("\nCertificate ARN:");
                System.out.println(certificateArn[0]);
```

```
            } else {
                Throwable cause = (ex instanceof CompletionException) ?
ex.getCause() : ex;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else {
                    System.err.println("Unexpected error: " + cause.getMessage());
                }
            }
        });

        future.join();
        return certificateArn[0];
    }
```

- Per i dettagli sull'API, CreateKeysAndCertificateconsulta 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.

```
    /**
     * Creates an IoT Thing with the specified name asynchronously.
     *
     * @param thingName The name of the IoT Thing to create.
     *
     * This method initiates an asynchronous request to create an IoT Thing with the
specified name.
     * If the request is successful, it prints the name of the thing and its ARN
value.
     * If an exception occurs, it prints the error message.
```

```
     */
    public void createIoTThing(String thingName) {
        CreateThingRequest createThingRequest = CreateThingRequest.builder()
            .thingName(thingName)
            .build();

        CompletableFuture<CreateThingResponse> future =
getAsyncClient().createThing(createThingRequest);
        future.whenComplete((createThingResponse, ex) -> {
            if (createThingResponse != null) {
                System.out.println(thingName + " was successfully created. The ARN
value is " + createThingResponse.thingArn());
            } else {
                Throwable cause = ex.getCause();
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else {
                    System.err.println("Unexpected error: " + cause.getMessage());
                }
            }
        });

        future.join();
    }
```

- Per i dettagli sull'API, CreateThingconsulta 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.

```
    /**
```

```
    * Creates an IoT rule asynchronously.
    *
    * @param roleARN The ARN of the IAM role that grants access to the rule's
actions.
    * @param ruleName The name of the IoT rule.
    * @param action The ARN of the action to perform when the rule is triggered.
    *
    * This method initiates an asynchronous request to create an IoT rule.
    * If the request is successful, it prints a confirmation message.
    * If an exception occurs, it prints the error message.
    */
   public void createIoTRule(String roleARN, String ruleName, String action) {
       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();

       CompletableFuture<CreateTopicRuleResponse> future =
getAsyncClient().createTopicRule(topicRuleRequest);
       future.whenComplete((response, ex) -> {
           if (response != null) {
               System.out.println("IoT Rule created successfully.");
           } else {
               Throwable cause = ex != null ? ex.getCause() : null;
               if (cause instanceof IotException) {
```

```
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else if (cause != null) {
                    System.err.println("Unexpected error: " + cause.getMessage());
                } else {
                    System.err.println("Failed to create IoT Rule.");
                }
            }
        });

        future.join();
    }
```

- Per i dettagli sull'API, CreateTopicRuleconsulta 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.

```
    /**
     * Deletes a certificate asynchronously.
     *
     * @param certificateArn The ARN of the certificate to delete.
     *
     * This method initiates an asynchronous request to delete a certificate.
     * If the deletion is successful, it prints a confirmation message.
     * If an exception occurs, it prints the error message.
     */
    public void deleteCertificate(String certificateArn) {
        DeleteCertificateRequest certificateProviderRequest =
DeleteCertificateRequest.builder()
            .certificateId(extractCertificateId(certificateArn))
            .build();
```

```
        CompletableFuture<DeleteCertificateResponse> future =
getAsyncClient().deleteCertificate(certificateProviderRequest);
        future.whenComplete((voidResult, ex) -> {
            if (ex == null) {
                System.out.println(certificateArn + " was successfully deleted.");
            } else {
                Throwable cause = ex.getCause();
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else {
                    System.err.println("Unexpected error: " + ex.getMessage());
                }
            }
        });

        future.join();
    }
```

- Per i dettagli sull'API, DeleteCertificateconsulta 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.

```
    /**
     * Deletes an IoT Thing asynchronously.
     *
     * @param thingName The name of the IoT Thing to delete.
     *
     * This method initiates an asynchronous request to delete an IoT Thing.
     * If the deletion is successful, it prints a confirmation message.
```

```
     * If an exception occurs, it prints the error message.
     */
    public void deleteIoTThing(String thingName) {
        DeleteThingRequest deleteThingRequest = DeleteThingRequest.builder()
            .thingName(thingName)
            .build();

        CompletableFuture<DeleteThingResponse> future =
getAsyncClient().deleteThing(deleteThingRequest);
        future.whenComplete((voidResult, ex) -> {
            if (ex == null) {
                System.out.println("Deleted Thing " + thingName);
            } else {
                Throwable cause = ex.getCause();
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else {
                    System.err.println("Unexpected error: " + ex.getMessage());
                }
            }
        });

        future.join();
    }
```

- Per i dettagli sull'API, DeleteThingconsulta 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.

```
    /**
```

```
     * Describes the endpoint of the IoT service asynchronously.
     *
     * @return A CompletableFuture containing the full endpoint URL.
     *
     * This method initiates an asynchronous request to describe the endpoint of the
 IoT service.
     * If the request is successful, it prints and returns the full endpoint URL.
     * If an exception occurs, it prints the error message.
     */
    public String describeEndpoint() {
        CompletableFuture<DescribeEndpointResponse> future =
 getAsyncClient().describeEndpoint(DescribeEndpointRequest.builder().endpointType("iot:Data-
ATS").build());
        final String[] result = {null};

        future.whenComplete((endpointResponse, ex) -> {
            if (endpointResponse != null) {
                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);
                result[0] = fullEndpoint;
            } else {
                Throwable cause = (ex instanceof CompletionException) ?
 ex.getCause() : ex;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
 cause).awsErrorDetails().errorMessage());
                } else {
                    System.err.println("Unexpected error: " + cause.getMessage());
                }
            }
        });

        future.join();
        return result[0];
    }
```

- Per i dettagli sull'API, DescribeEndpointconsulta 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.

```
    /**
     * Describes an IoT Thing asynchronously.
     *
     * @param thingName The name of the IoT Thing.
     *
     * This method initiates an asynchronous request to describe an IoT Thing.
     * If the request is successful, it prints the Thing details.
     * If an exception occurs, it prints the error message.
     */
    private void describeThing(String thingName) {
        DescribeThingRequest thingRequest = DescribeThingRequest.builder()
            .thingName(thingName)
            .build();

        CompletableFuture<DescribeThingResponse> future =
getAsyncClient().describeThing(thingRequest);
        future.whenComplete((describeResponse, ex) -> {
            if (describeResponse != null) {
                System.out.println("Thing Details:");
                System.out.println("Thing Name: " + describeResponse.thingName());
                System.out.println("Thing ARN: " + describeResponse.thingArn());
            } else {
                Throwable cause = ex != null ? ex.getCause() : null;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else if (cause != null) {
                    System.err.println("Unexpected error: " + cause.getMessage());
                } else {
                    System.err.println("Failed to describe Thing.");
                }
```

```
            }
        });

        future.join();
    }
```

- Per i dettagli sull'API, DescribeThingconsulta 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.

```
    /**
     * Detaches a principal (certificate) from an IoT Thing asynchronously.
     *
     * @param thingName The name of the IoT Thing.
     * @param certificateArn The ARN of the certificate to detach.
     *
     * This method initiates an asynchronous request to detach a certificate from an
IoT Thing.
     * If the detachment is successful, it prints a confirmation message.
     * If an exception occurs, it prints the error message.
     */
    public void detachThingPrincipal(String thingName, String certificateArn) {
        DetachThingPrincipalRequest thingPrincipalRequest =
DetachThingPrincipalRequest.builder()
            .principal(certificateArn)
            .thingName(thingName)
            .build();

        CompletableFuture<DetachThingPrincipalResponse> future =
getAsyncClient().detachThingPrincipal(thingPrincipalRequest);
```

```
        future.whenComplete((voidResult, ex) -> {
            if (ex == null) {
                System.out.println(certificateArn + " was successfully removed from
" + thingName);
            } else {
                Throwable cause = ex.getCause();
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else {
                    System.err.println("Unexpected error: " + ex.getMessage());
                }
            }
        });

        future.join();
    }
```

- Per i dettagli sull'API, DetachThingPrincipalconsulta 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.

```
    /**
     * Lists all certificates asynchronously.
     *
     * This method initiates an asynchronous request to list all certificates.
     * If the request is successful, it prints the certificate IDs and ARNs.
     * If an exception occurs, it prints the error message.
     */
    public void listCertificates() {
```

```
        CompletableFuture<ListCertificatesResponse> future =
getAsyncClient().listCertificates();
        future.whenComplete((response, ex) -> {
            if (response != null) {
                List<Certificate> certList = response.certificates();
                for (Certificate cert : certList) {
                    System.out.println("Cert id: " + cert.certificateId());
                    System.out.println("Cert Arn: " + cert.certificateArn());
                }
            } else {
                Throwable cause = ex != null ? ex.getCause() : null;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else if (cause != null) {
                    System.err.println("Unexpected error: " + cause.getMessage());
                } else {
                    System.err.println("Failed to list certificates.");
                }
            }
        });

        future.join();
    }
```

- Per i dettagli sull'API, ListCertificatesconsulta 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.

```
    /**
```

```
     * Searches for IoT Things asynchronously based on a query string.
     *
     * @param queryString The query string to search for Things.
     *
     * This method initiates an asynchronous request to search for IoT Things.
     * If the request is successful and Things are found, it prints their IDs.
     * If no Things are found, it prints a message indicating so.
     * If an exception occurs, it prints the error message.
     */
    public void searchThings(String queryString) {
        SearchIndexRequest searchIndexRequest = SearchIndexRequest.builder()
            .queryString(queryString)
            .build();

        CompletableFuture<SearchIndexResponse> future =
getAsyncClient().searchIndex(searchIndexRequest);
        future.whenComplete((searchIndexResponse, ex) -> {
            if (searchIndexResponse != null) {
                // 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()));
                }
            } else {
                Throwable cause = ex != null ? ex.getCause() : null;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else if (cause != null) {
                    System.err.println("Unexpected error: " + cause.getMessage());
                } else {
                    System.err.println("Failed to search for IoT Things.");
                }
            }
        });

        future.join();
    }
```

- Per i dettagli sull'API, SearchIndexconsulta 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.

```
   /**
    * Updates the shadow of an IoT Thing asynchronously.
    *
    * @param thingName The name of the IoT Thing.
    *
    * This method initiates an asynchronous request to update the shadow of an IoT
Thing.
    * If the request is successful, it prints a confirmation message.
    * If an exception occurs, it prints the error message.
    */
   public void updateShadowThing(String thingName) {
       // 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();

       CompletableFuture<UpdateThingShadowResponse> future =
getAsyncDataPlaneClient().updateThingShadow(updateThingShadowRequest);
       future.whenComplete((updateResponse, ex) -> {
           if (updateResponse != null) {
               System.out.println("Thing Shadow updated successfully.");
           } else {
               Throwable cause = ex != null ? ex.getCause() : null;
               if (cause instanceof IotException) {
                   System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
```

```
                } else if (cause != null) {
                    System.err.println("Unexpected error: " + cause.getMessage());
                } else {
                    System.err.println("Failed to update Thing Shadow.");
                }
            }
        });

        future.join();
    }
```

- Per i dettagli sull'API, UpdateThingconsulta AWS SDK for Java 2.x API Reference.

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.

Ogni esempio include un collegamento al codice sorgente completo, in cui 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.

```
   /**
    * Retrieves the payload of a Thing's shadow asynchronously.
    *
    * @param thingName The name of the IoT Thing.
    *
    * This method initiates an asynchronous request to get the payload of a Thing's
shadow.
    * If the request is successful, it prints the shadow data.
    * If an exception occurs, it prints the error message.
    */
   public void getPayload(String thingName) {
       GetThingShadowRequest getThingShadowRequest =
GetThingShadowRequest.builder()
           .thingName(thingName)
           .build();

       CompletableFuture<GetThingShadowResponse> future =
getAsyncDataPlaneClient().getThingShadow(getThingShadowRequest);
       future.whenComplete((getThingShadowResponse, ex) -> {
           if (getThingShadowResponse != null) {
               // Extracting payload from response.
               SdkBytes payload = getThingShadowResponse.payload();
               String payloadString = payload.asUtf8String();
               System.out.println("Received Shadow Data: " + payloadString);
           } else {
               Throwable cause = ex != null ? ex.getCause() : null;
               if (cause instanceof IotException) {
                   System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
               } else if (cause != null) {
                   System.err.println("Unexpected error: " + cause.getMessage());
               } else {
                   System.err.println("Failed to get Thing Shadow payload.");
               }
```

```
            }
        });

        future.join();
    }
```

- Per i dettagli sull'API, GetThingShadowconsulta 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.

```
    /**
     * Updates the shadow of an IoT Thing asynchronously.
     *
     * @param thingName The name of the IoT Thing.
     *
     * This method initiates an asynchronous request to update the shadow of an IoT
Thing.
     * If the request is successful, it prints a confirmation message.
     * If an exception occurs, it prints the error message.
     */
    public void updateShadowThing(String thingName) {
        // 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();
```

```
        CompletableFuture<UpdateThingShadowResponse> future =
getAsyncDataPlaneClient().updateThingShadow(updateThingShadowRequest);
        future.whenComplete((updateResponse, ex) -> {
            if (updateResponse != null) {
                System.out.println("Thing Shadow updated successfully.");
            } else {
                Throwable cause = ex != null ? ex.getCause() : null;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else if (cause != null) {
                    System.err.println("Unexpected error: " + cause.getMessage());
                } else {
                    System.err.println("Failed to update Thing Shadow.");
                }
            }
        });

        future.join();
    }
```

- Per i dettagli sull'API, UpdateThingShadowconsulta AWS SDK for Java 2.x API Reference.

AWS IoT FleetWise 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 FleetWise.

Le nozioni di base sono esempi di codice che mostrano come eseguire le operazioni essenziali all'interno di un servizio.

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.

Ogni esempio include un collegamento al codice sorgente completo, in cui è possibile trovare istruzioni su come configurare ed eseguire il codice nel contesto.

Nozioni di base

Salve AWS IoT FleetWise

L'esempio di codice seguente mostra come iniziare a utilizzare AWS IoT FleetWise.

SDK per Java 2.x

> **Note**
>
> C'è 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 HelloFleetwise {

        public static void main(String[] args) {
            ListSignalCatalogs();
        }

        public static void  ListSignalCatalogs() {
            try (IoTFleetWiseClient fleetWiseClient = IoTFleetWiseClient.builder()
                    .region(Region.US_EAST_1)
                    .credentialsProvider(DefaultCredentialsProvider.create())
                    .build()) {

                ListSignalCatalogsRequest request =
 ListSignalCatalogsRequest.builder()
                        .maxResults(10) // Optional: limit per page
                        .build();

                ListSignalCatalogsIterable paginator =
 fleetWiseClient.listSignalCatalogsPaginator(request);
                boolean found = false;

                for (ListSignalCatalogsResponse response : paginator) {
                    for (SignalCatalogSummary summary : response.summaries()) {
                        found = true;
                        System.out.println("Catalog Name: " + summary.name());
                        System.out.println("ARN: " + summary.arn());
                        System.out.println("Created: " + summary.creationTime());
                        System.out.println("Last Modified: " +
 summary.lastModificationTime());
                        System.out.println("---------------");
                    }
```

```
                }

                if (!found) {
                    System.out.println("No AWS Fleetwise Signal Catalogs were
found.");
                }

            } catch (IoTFleetWiseException e) {
                System.err.println("Error listing signal catalogs: " +
e.awsErrorDetails().errorMessage());
                throw new RuntimeException(e);
            }
        }
    }
```

- Per i dettagli sulle API, consulta listSignalCatalogsPaginator in AWS SDK for Java 2.x API Reference.

Argomenti

- Nozioni di base
- Azioni

Nozioni di base

Informazioni di base

L'esempio di codice seguente mostra come:

- Crea una raccolta di segnali standardizzati.
- Crea una flotta che rappresenti un gruppo di veicoli.
- Crea un manifesto del modello.
- Crea un manifesto del decodificatore.
- Controllate lo stato del manifesto del modello.
- Controllate lo stato del decoder.
- Crea un oggetto IoT.
- Crea un veicolo.

- Visualizza i dettagli del veicolo.
- Eliminare gli AWS IoT FleetWise asset.

SDK per Java 2.x

> **Note**
>
> C'è altro da fare 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 AWS IoT SiteWise funzionalità.

```
public class FleetwiseScenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");
    static FleetwiseActions actions = new FleetwiseActions();
    private static final Logger logger =
 LoggerFactory.getLogger(FleetwiseScenario.class);
    static Scanner scanner = new Scanner(System.in);

    public static void main(String[] args) {
        final String usage =
                """
                Usage:
                    <signalCatalogName> <manifestName> <fleetId> <vecName> <decName>

                Where:
                    signalCatalogName     - The name of the Signal Catalog to create
 (eg, catalog30).
                    manifestName          - The name of the Vehicle Model (Model
 Manifest) to create (eg, manifest30).
                    fleetId               - The ID of the Fleet to create (eg,
 fleet30).
                    vecName               - The name of the Vehicle to create (eg,
 vehicle30).
                    decName               - The name of the Decoder Manifest to
 create (eg, decManifest30).
                """;

       if (args.length != 5) {
           logger.info(usage);
```

```
            return;
        }

        String signalCatalogName = args[0];
        String manifestName = args[1];
        String fleetId = args[2];
        String vecName = args[3];
        String decName = args[4];

        logger.info(
                """
                 AWS IoT FleetWise is a managed service that simplifies the
                 process of collecting, organizing, and transmitting vehicle
                 data to the cloud in near real-time. Designed for automakers
                 and fleet operators, it allows you to define vehicle models,
                 specify the exact data you want to collect (such as engine
                 temperature, speed, or battery status), and send this data to
                 AWS for analysis. By using intelligent data collection
                 techniques, IoT FleetWise reduces the volume of data
                 transmitted by filtering and transforming it at the edge,
                 helping to minimize bandwidth usage and costs.

                At its core, AWS IoT FleetWise helps organizations build
                scalable systems for vehicle data management and analytics,
                supporting a wide variety of vehicles and sensor configurations.
                You can define signal catalogs and decoder manifests that describe
                how raw CAN bus signals are translated into readable data, making
                the platform highly flexible and extensible. This allows
                manufacturers to optimize vehicle performance, improve safety,
                and reduce maintenance costs by gaining real-time visibility
                into fleet operations.
                """);

        waitForInputToContinue(scanner);
        logger.info(DASHES);
        try {
            runScenario(signalCatalogName, manifestName, fleetId, vecName, decName);
        } catch (RuntimeException e) {
            logger.info(e.getMessage());
        }
    }

    private static void runScenario(String signalCatalogName,
                                    String manifestName,
```

```
                                    String fleetId,
                                    String vecName,
                                    String decName) {
        logger.info(DASHES);
        logger.info("1. Creates a collection of standardized signals that can be
 reused to create vehicle models");
        String signalCatalogArn;
        try {
            signalCatalogArn =
 actions.createSignalCatalogAsync(signalCatalogName).join();
            logger.info("The collection ARN is " + signalCatalogArn);
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ValidationException) {
                logger.error("The request failed due to a validation issue: {}",
 cause.getMessage());
            } else {
                logger.error("An unexpected error occurred.", cause);
            }
            return;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("2. Create a fleet that represents a group of vehicles");
        logger.info(
                """
                Creating an IoT FleetWise fleet allows you to efficiently collect,
                organize, and transfer vehicle data to the cloud, enabling real-
time
                insights into vehicle performance and health.

                It helps reduce data costs by allowing you to filter and prioritize
                only the most relevant vehicle signals, supporting advanced
 analytics
                and predictive maintenance use cases.
                """);

        waitForInputToContinue(scanner);
        String fleetid;
        try {
            fleetid = actions.createFleetAsync(signalCatalogArn, fleetId).join();
            logger.info("The fleet Id is " + fleetid);
```

```
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ResourceNotFoundException) {
                logger.error("The resource was not found: {}", cause.getMessage());
            } else {
                logger.error("An unexpected error occurred.", cause);
            }
            return;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("3. Create a model manifest");
        logger.info(
                """
                An AWS IoT FleetWise manifest defines the structure and
                relationships of vehicle data. The model manifest specifies
                which signals to collect and how they relate to vehicle systems,
                while the decoder manifest defines how to decode raw vehicle data
                into meaningful signals.
                """);
        waitForInputToContinue(scanner);
        String manifestArn;
        try {
            List<Node> nodes =
actions.listSignalCatalogNodeAsync(signalCatalogName).join();
            manifestArn = actions.createModelManifestAsync(manifestName,
signalCatalogArn, nodes).join();
            logger.info("The manifest ARN is {}", manifestArn);
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            logger.error("An unexpected error occurred.", cause);
            return;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("4. Create a decoder manifest");
        logger.info(
                """
                A decoder manifest in AWS IoT FleetWise defines how raw vehicle
                data (such as CAN signals) should be interpreted and decoded
```

```
                into meaningful signals. It acts as a translation layer
                that maps vehicle-specific protocols to standardized data formats
                using decoding rules. This is crucial for extracting usable
                data from different vehicle models, even when their data
                formats vary.

                """);
        waitForInputToContinue(scanner);
        String decArn;
        try {
            decArn = actions.createDecoderManifestAsync(decName,
manifestArn).join();
            logger.info("The decoder manifest ARN is {}", decArn);
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            logger.error("An unexpected error occurred.", cause);
            return;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info("5. Check the status of the model manifest");
        logger.info(
                """
                The model manifest must be in an ACTIVE state before it can be used
                to create or update a vehicle.
                """);
        waitForInputToContinue(scanner);
        try {
            actions.updateModelManifestAsync(manifestName);
            actions.waitForModelManifestActiveAsync(manifestName).join();
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            logger.error("An unexpected error occurred while waiting for the model
manifest status.", cause);
            return;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info("6. Check the status of the decoder");
        logger.info(
                """
```

```
                The decoder manifest must be in an ACTIVE state before it can be
used
                to create or update a vehicle.
                """);
        waitForInputToContinue(scanner);
        try {
            actions.updateDecoderManifestAsync(decName);
            actions.waitForDecoderManifestActiveAsync(decName).join();
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            logger.error("An unexpected error occurred while waiting for the decoder
manifest status.", cause);
            return;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("7. Create an IoT Thing");
        logger.info(
                """
                AWS IoT FleetWise expects an existing AWS IoT Thing with the same
                name as the vehicle name you are passing to createVehicle method.
                Before calling createVehicle(), you must create an AWS IoT Thing
                with the same name using the AWS IoT Core service.
                """);
        waitForInputToContinue(scanner);
        try {
            actions.createThingIfNotExistsAsync(vecName).join();
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ResourceAlreadyExistsException) {
                logger.error("The resource exists: {}", cause.getMessage());
            } else {
                logger.error("An unexpected error occurred.", cause);
                return;
            }
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("8. Create a vehicle");
        logger.info(
```

```
                """
                Creating a vehicle in AWS IoT FleetWise allows you to digitally
                represent and manage a physical vehicle within the AWS ecosystem.
                This enables efficient ingestion, transformation, and transmission
                of vehicle telemetry data to the cloud for analysis.
                """);
        waitForInputToContinue(scanner);
        try {
            actions.createVehicleAsync(vecName, manifestArn, decArn).join();
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();

            if (cause instanceof ResourceNotFoundException) {
                logger.error("The required resource was not found: {}",
cause.getMessage());
            } else {
                logger.error("An unexpected error occurred while creating vehicle.",
cause);
            }
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("9. Display vehicle details");
        waitForInputToContinue(scanner);
        try {
            actions.getVehicleDetailsAsync(vecName).join();
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ResourceNotFoundException) {
                logger.error("The resource was not found: {}", cause.getMessage());
            } else {
                logger.error("An unexpected error occurred.", cause);
            }
            return;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("10. Delete the AWS IoT Fleetwise Assets");
        logger.info("Would you like to delete the IoT Fleetwise Assets? (y/n)");
        String delAns = scanner.nextLine().trim();
```

```
        if (delAns.equalsIgnoreCase("y")) {
            try {
                actions.deleteVehicleAsync(vecName).join();
                actions.deleteDecoderManifestAsync(decName).join();
                actions.deleteModelManifestAsync(manifestName).join();
                actions.deleteFleetAsync(fleetid).join();
                actions.deleteSignalCatalogAsync(signalCatalogName).join();
            } catch (CompletionException ce) {
                Throwable cause = ce.getCause();
                if (cause instanceof ResourceNotFoundException) {
                    // Handle the case where the resource is not found.
                    logger.error("The resource was not found: {}",
 cause.getMessage());
                } else if (cause instanceof RuntimeException) {
                    // Handle other runtime exceptions.
                    logger.error("An unexpected error occurred: {}",
 cause.getMessage());
                } else {
                    // Catch any other unexpected exceptions.
                    logger.error("An unknown error occurred.", cause);
                }
                return;
            }

            logger.info(DASHES);
            logger.info(
                    """
                    Thank you for checking out the AWS IoT Fleetwise Service Use
 demo. We hope you
                    learned something new, or got some inspiration for your own apps
 today.
                    For more AWS code examples, have a look at:
                    https://docs.aws.amazon.com/code-library/latest/ug/what-is-code-
library.html
                    """);
            logger.info(DASHES);
        } else {
            logger.info("The AWS resources will not be deleted.");
        }
    }

    private static void waitForInputToContinue(Scanner scanner) {
        while (true) {
            logger.info("");
```

```
            logger.info("Enter 'c' followed by <ENTER> to continue:");
            String input = scanner.nextLine();

            if (input.trim().equalsIgnoreCase("c")) {
                logger.info("Continuing with the program...");
                logger.info("");
                break;
            } else {
                logger.info("Invalid input. Please try again.");
            }
        }
    }
}
```

Una classe wrapper per i metodi SDK. AWS IoT FleetWise

```
public class FleetwiseActions {
    private static final Logger logger =
 LoggerFactory.getLogger(FleetwiseActions.class);
    private static IoTFleetWiseAsyncClient ioTFleetWiseAsyncClient;

    private static IoTFleetWiseAsyncClient getAsyncClient() {
        if (ioTFleetWiseAsyncClient == null) {
            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                    .maxConcurrency(100)
                    .connectionTimeout(Duration.ofSeconds(60))
                    .readTimeout(Duration.ofSeconds(60))
                    .writeTimeout(Duration.ofSeconds(60))
                    .build();

            ClientOverrideConfiguration overrideConfig =
 ClientOverrideConfiguration.builder()
                    .apiCallTimeout(Duration.ofMinutes(2))
                    .apiCallAttemptTimeout(Duration.ofSeconds(90))
                    .retryStrategy(RetryMode.STANDARD)
                    .build();

            ioTFleetWiseAsyncClient = IoTFleetWiseAsyncClient.builder()
                    .httpClient(httpClient)
                    .overrideConfiguration(overrideConfig)
                    .build();
        }
```

```
        return ioTFleetWiseAsyncClient;
    }


    /**
     * Creates a signal catalog.
     *
     * @param signalCatalogName the name of the signal catalog to be created
     * @return a {@link CompletableFuture} that completes with the Amazon Resource
Name (ARN) of the created signal catalog
     */
    public CompletableFuture<String> createSignalCatalogAsync(String
signalCatalogName) {
        return deleteSignalCatalogIfExistsAsync(signalCatalogName)
                .thenCompose(ignored -> delayAsync(2000)) // Wait for 2 seconds
                .thenCompose(ignored -> {
                    List<Node> nodes = List.of(
                            Node.builder().branch(
                                    Branch.builder()
                                            .fullyQualifiedName("Vehicle")
                                            .description("Root branch")
                                            .build()
                            ).build(),
                            Node.builder().branch(
                                    Branch.builder()

.fullyQualifiedName("Vehicle.Powertrain")
                                            .description("Powertrain branch")
                                            .build()
                            ).build(),
                            Node.builder().sensor(
                                    Sensor.builder()

.fullyQualifiedName("Vehicle.Powertrain.EngineRPM")
                                            .description("Engine RPM")
                                            .dataType(NodeDataType.DOUBLE)
                                            .unit("rpm")
                                            .build()
                            ).build(),
                            Node.builder().sensor(
                                    Sensor.builder()

.fullyQualifiedName("Vehicle.Powertrain.VehicleSpeed")
                                            .description("Vehicle Speed")
```

```
                                        .dataType(NodeDataType.DOUBLE)
                                        .unit("km/h")
                                        .build()
                        ).build()
                );

                CreateSignalCatalogRequest request =
CreateSignalCatalogRequest.builder()
                        .name(signalCatalogName)
                        .nodes(nodes)
                        .build();

                CompletableFuture<String> result = new CompletableFuture<>();

                getAsyncClient().createSignalCatalog(request)
                        .whenComplete((response, exception) -> {
                            if (exception != null) {
                                Throwable cause = exception.getCause() != null ?
exception.getCause() : exception;

                                if (cause instanceof ValidationException) {
                                    result.completeExceptionally(cause);
                                } else {
                                    result.completeExceptionally(new
RuntimeException("Error creating the catalog", cause));
                                }
                            } else {
                                result.complete(response.arn());
                            }
                        });

                return result;
            });
    }

    /**
     * Delays the execution of the current thread asynchronously for the specified
duration.
     *
     * @param millis the duration of the delay in milliseconds
     * @return a {@link CompletableFuture} that completes after the specified delay
     */
    private static CompletableFuture<Void> delayAsync(long millis) {
        return CompletableFuture.runAsync(() -> {
```

```
            try {
                Thread.sleep(millis);
            } catch (InterruptedException e) {
                throw new CompletionException("Sleep interrupted", e);
            }
        });
    }

    /**
     * Deletes the specified signal catalog.
     *
     * @param signalCatalogName the name of the signal catalog to delete
     * @return a {@link CompletableFuture} representing the asynchronous operation.
     */
    public static CompletableFuture<Void> deleteSignalCatalogIfExistsAsync(String
signalCatalogName) {
        DeleteSignalCatalogRequest request = DeleteSignalCatalogRequest.builder()
                .name(signalCatalogName)
                .build();

        return getAsyncClient().deleteSignalCatalog(request)
                .handle((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause() != null ?
exception.getCause() : exception;
                        if (cause instanceof ResourceNotFoundException) {
                           throw new CompletionException(new
RuntimeException("Signal Catalog not found: " + signalCatalogName));
                        }
                        throw new RuntimeException("Failed to delete signal catalog:
" + signalCatalogName, cause);
                    }
                    return null;
                });
    }


    /**
     * Creates a new decoder manifest.
     *
     * @param name             the name of the decoder manifest
     * @param modelManifestArn the ARN of the model manifest
     * @return a {@link CompletableFuture} that completes with the ARN of the
created decoder manifest
```

```
     */
    public CompletableFuture<String> createDecoderManifestAsync(String name, String
modelManifestArn) {
        String interfaceId = "can0";
        NetworkInterface networkInterface = NetworkInterface.builder()
                .interfaceId(interfaceId)
                .type(NetworkInterfaceType.CAN_INTERFACE)
                .canInterface(CanInterface.builder()
                        .name("canInterface0")
                        .protocolName("CAN")
                        .protocolVersion("1.0")
                        .build())
                .build();

        // Vehicle.Powertrain.EngineRPM decoder.
        SignalDecoder engineRpmDecoder = SignalDecoder.builder()
                .fullyQualifiedName("Vehicle.Powertrain.EngineRPM")
                .interfaceId(interfaceId)
                .type(SignalDecoderType.CAN_SIGNAL)
                .canSignal(CanSignal.builder()
                        .messageId(100)
                        .isBigEndian(false)
                        .isSigned(false)
                        .startBit(0)
                        .length(16)
                        .factor(1.0)
                        .offset(0.0)
                        .build())
                .build();

        // Vehicle.Powertrain.VehicleSpeed decoder.
        SignalDecoder vehicleSpeedDecoder = SignalDecoder.builder()
                .fullyQualifiedName("Vehicle.Powertrain.VehicleSpeed")
                .interfaceId(interfaceId)
                .type(SignalDecoderType.CAN_SIGNAL)
                .canSignal(CanSignal.builder()
                        .messageId(101)
                        .isBigEndian(false)
                        .isSigned(false)
                        .startBit(16)
                        .length(16)
                        .factor(1.0)
                        .offset(0.0)
                        .build())
```

```
                .build();

        CreateDecoderManifestRequest request =
CreateDecoderManifestRequest.builder()
                .name(name)
                .modelManifestArn(modelManifestArn)
                .networkInterfaces(List.of(networkInterface))
                .signalDecoders(List.of(engineRpmDecoder, vehicleSpeedDecoder))
                .build();

        CompletableFuture<String> result = new CompletableFuture<>();

        getAsyncClient().createDecoderManifest(request)
                .whenComplete((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause() != null ?
exception.getCause() : exception;

                        if (cause instanceof DecoderManifestValidationException) {
                            result.completeExceptionally(new
CompletionException("The request contains signal decoders with validation errors: "
+ cause.getMessage(), cause));
                        } else {
                            result.completeExceptionally(new
CompletionException("Failed to create decoder manifest: " + exception.getMessage(),
exception));
                        }
                    } else {
                        result.complete(response.arn()); // Complete successfully
with the ARN
                    }
                });

        return result;
    }

    /**
     * Deletes a decoder manifest.
     *
     * @param name the name of the decoder manifest to delete
     * @return a {@link CompletableFuture} that completes when the decoder manifest
has been deleted
     */
    public CompletableFuture<Void> deleteDecoderManifestAsync(String name) {
```

```
        return
getAsyncClient().deleteDecoderManifest(DeleteDecoderManifestRequest.builder().name(name).bu
                .handle((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause() != null ?
exception.getCause() : exception;
                        if (cause instanceof ResourceNotFoundException) {
                            throw (ResourceNotFoundException) cause;
                        }
                        throw new RuntimeException("Failed to delete the decoder
manifest: " + cause);
                    }
                    return null;
                });
    }


    /**
     * Deletes a vehicle with the specified name.
     *
     * @param vecName the name of the vehicle to be deleted
     * @return a {@link CompletableFuture} that completes when the vehicle has been
deleted
     */
    public CompletableFuture<Void> deleteVehicleAsync(String vecName) {
        DeleteVehicleRequest request = DeleteVehicleRequest.builder()
                .vehicleName(vecName)
                .build();

        return getAsyncClient().deleteVehicle(request)
                .handle((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause() != null ?
exception.getCause() : exception;
                        if (cause instanceof ResourceNotFoundException) {
                            throw (ResourceNotFoundException) cause;
                        }
                        throw new RuntimeException("Failed to delete the vehicle: "
+ cause);
                    }
                    return null;
                });
    }
```

```
   /**
    * Updates the model manifest.
    *
    * @param name the name of the model manifest to update
    */
   public void updateModelManifestAsync(String name) {
       UpdateModelManifestRequest request = UpdateModelManifestRequest.builder()
               .name(name)
               .status(ManifestStatus.ACTIVE)
               .build();

       getAsyncClient().updateModelManifest(request)
               .whenComplete((response, exception) -> {
                   if (exception != null) {
                       throw new CompletionException("Failed to update model
manifest: " + exception.getMessage(), exception);
                   }
               })
               .thenApply(response -> null);
   }


   /**
    * Updates the decoder manifest with the given name.
    *
    * @param name the name of the decoder manifest to update
    * @return a {@link CompletableFuture} that completes when the update operation
is finished
    */
   public CompletableFuture<Void> updateDecoderManifestAsync(String name) {
       UpdateDecoderManifestRequest request =
UpdateDecoderManifestRequest.builder()
               .name(name)
               .status(ManifestStatus.ACTIVE)
               .build();

       return getAsyncClient().updateDecoderManifest(request)
               .whenComplete((response, exception) -> {
                   if (exception != null) {
                       throw new CompletionException("Failed to update decoder
manifest: " + exception.getMessage(), exception);
                   }
               })
```

```
                .thenApply(response -> null);
    }


    /**
     * Creates a new vehicle in the system.
     *
     * @param vecName     the name of the vehicle to be created
     * @param manifestArn the Amazon Resource Name (ARN) of the model manifest for
the vehicle
     * @param decArn      the Amazon Resource Name (ARN) of the decoder manifest for
the vehicle
     * @return a {@link CompletableFuture} that completes when the vehicle has been
created, or throws a
     */
    public CompletableFuture<Void> createVehicleAsync(String vecName, String
manifestArn, String decArn) {
        CreateVehicleRequest request = CreateVehicleRequest.builder()
                .vehicleName(vecName)
                .modelManifestArn(manifestArn)
                .decoderManifestArn(decArn)
                .build();

        CompletableFuture<Void> result = new CompletableFuture<>();
        getAsyncClient().createVehicle(request)
                .whenComplete((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception instanceof CompletionException ?
exception.getCause() : exception;

                        if (cause instanceof ResourceNotFoundException) {
                            result.completeExceptionally(cause);
                        } else {
                            result.completeExceptionally(new
RuntimeException("Failed to create vehicle: " + cause.getMessage(), cause));
                        }
                    } else {
                        logger.info("Vehicle '{}' created successfully.", vecName);
                        result.complete(null); // mark future as complete
                    }
                });

        return result;
    }
```

```
   /**
    * Waits for the decoder manifest to become active.
    *
    * @param decoderName the name of the decoder to wait for
    * @return a {@link CompletableFuture} that completes when the decoder manifest
becomes active, or exceptionally if an error occurs or the manifest becomes invalid
    */
   public CompletableFuture<Void> waitForDecoderManifestActiveAsync(String
decoderName) {
       CompletableFuture<Void> result = new CompletableFuture<>();

       ScheduledExecutorService scheduler =
Executors.newSingleThreadScheduledExecutor();
       AtomicInteger secondsElapsed = new AtomicInteger(0);
       AtomicReference<ManifestStatus> lastStatus = new
AtomicReference<>(ManifestStatus.DRAFT);

       logger.info(" Elapsed: 0s | Decoder Status: DRAFT");

       final Runnable pollTask = new Runnable() {
           @Override
           public void run() {
               int elapsed = secondsElapsed.incrementAndGet();

               // Check status every 5 seconds
               if (elapsed % 5 == 0) {
                   GetDecoderManifestRequest request =
GetDecoderManifestRequest.builder()
                           .name(decoderName)
                           .build();

                   getAsyncClient().getDecoderManifest(request)
                           .whenComplete((response, exception) -> {
                               if (exception != null) {
                                   Throwable cause = exception instanceof
CompletionException ? exception.getCause() : exception;

                                   scheduler.shutdown();
                                   if (cause instanceof ResourceNotFoundException)
{
                                       result.completeExceptionally(new
RuntimeException("Decoder manifest not found: " + cause.getMessage(), cause));
```

```
                                    } else {
                                        result.completeExceptionally(new
RuntimeException("Error while polling decoder manifest status: " +
exception.getMessage(), exception));
                                    }
                                    return;
                                }

                                ManifestStatus status = response.status();
                                lastStatus.set(status);

                                if (status == ManifestStatus.ACTIVE) {
                                    logger.info("\r Elapsed: {}s | Decoder Status:
ACTIVE", elapsed);
                                    scheduler.shutdown();
                                    result.complete(null);
                                } else if (status == ManifestStatus.INVALID) {
                                    logger.info("\r Elapsed: {}s | Decoder Status:
INVALID", elapsed);
                                    scheduler.shutdown();
                                    result.completeExceptionally(new
RuntimeException("Decoder manifest became INVALID. Cannot proceed."));
                                } else {
                                    logger.info("\r# Elapsed: {}s | Decoder Status:
{}", elapsed, status);
                                }
                            });
                } else {
                    logger.info("\r Elapsed: {}s | Decoder Status: {}", elapsed,
lastStatus.get());
                }
            }
        };

        // Start the task with an initial delay of 1 second, and repeat every second
        scheduler.scheduleAtFixedRate(pollTask, 1, 1, TimeUnit.SECONDS);
        return result;
    }



    /**
     * Waits for the specified model manifest to become active.
     *
```

```
     * @param manifestName the name of the model manifest to wait for
     */
    public CompletableFuture<Void> waitForModelManifestActiveAsync(String
manifestName) {
        CompletableFuture<Void> result = new CompletableFuture<>();

        ScheduledExecutorService scheduler =
Executors.newSingleThreadScheduledExecutor();
        AtomicInteger secondsElapsed = new AtomicInteger(0);
        AtomicReference<ManifestStatus> lastStatus = new
AtomicReference<>(ManifestStatus.DRAFT);

        logger.info("Elapsed: 0s | Status: DRAFT");

        final Runnable pollTask = new Runnable() {
            @Override
            public void run() {
                int elapsed = secondsElapsed.incrementAndGet();

                // Only check status every 5 seconds
                if (elapsed % 5 == 0) {
                    GetModelManifestRequest request =
GetModelManifestRequest.builder()
                            .name(manifestName)
                            .build();

                    getAsyncClient().getModelManifest(request)
                            .whenComplete((response, exception) -> {
                                if (exception != null) {
                                    Throwable cause = exception instanceof
CompletionException ? exception.getCause() : exception;

                                    scheduler.shutdown();
                                    if (cause instanceof ResourceNotFoundException)
{
                                        result.completeExceptionally(new
RuntimeException("Model manifest not found: " + cause.getMessage(), cause));
                                    } else {
                                        result.completeExceptionally(new
RuntimeException("Error while polling model manifest status: " +
exception.getMessage(), exception));
                                    }
                                    return;
                                }
```

```
                                ManifestStatus status = response.status();
                                lastStatus.set(status);

                                if (status == ManifestStatus.ACTIVE) {
                                    logger.info("\rElapsed: {}s | Status: ACTIVE",
elapsed);
                                    scheduler.shutdown();
                                    result.complete(null);
                                } else if (status == ManifestStatus.INVALID) {
                                    logger.info("\rElapsed: {}s | Status: INVALID",
elapsed);
                                    scheduler.shutdown();
                                    result.completeExceptionally(new
RuntimeException("Model manifest became INVALID. Cannot proceed."));
                                } else {
                                    logger.info("\rElapsed: {}s | Status: {}",
elapsed, status);
                                }
                            });
                } else {
                    logger.info("\rElapsed: {}s | Status: {}", elapsed,
lastStatus.get());
                }
            }
        };

        // Start the task with an initial delay of 1 second, and repeat every second
        scheduler.scheduleAtFixedRate(pollTask, 1, 1, TimeUnit.SECONDS);
        return result;
    }



    /**
     * Fetches the details of a vehicle.
     *
     * @param vehicleName the name of the vehicle to fetch details for
     * @return a {@link CompletableFuture} that completes when the vehicle details
have been fetched
     */
    public CompletableFuture<Void> getVehicleDetailsAsync(String vehicleName) {
        GetVehicleRequest request = GetVehicleRequest.builder()
                .vehicleName(vehicleName)
```

```
                .build();

        CompletableFuture<Void> result = new CompletableFuture<>();

        getAsyncClient().getVehicle(request)
                .whenComplete((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception instanceof CompletionException ?
exception.getCause() : exception;

                        if (cause instanceof ResourceNotFoundException) {
                            result.completeExceptionally(cause); // don't rewrap
                        } else {
                            result.completeExceptionally(new
RuntimeException("Failed to fetch vehicle details: " + cause.getMessage(), cause));
                        }
                    } else {
                        Map<String, Object> details = new HashMap<>();
                        details.put("vehicleName", response.vehicleName());
                        details.put("arn", response.arn());
                        details.put("modelManifestArn",
response.modelManifestArn());
                        details.put("decoderManifestArn",
response.decoderManifestArn());
                        details.put("attributes", response.attributes());
                        details.put("creationTime",
response.creationTime().toString());
                        details.put("lastModificationTime",
response.lastModificationTime().toString());

                        logger.info("Vehicle Details:");
                        details.forEach((key, value) -> logger.info("• {} : {}",
key, value));

                        result.complete(null); // mark as successful
                    }
                });

        return result;
    }


    /**
     * Creates an IoT Thing if it does not already exist.
```

```
     *
     * @param thingName the name of the IoT Thing to create
     * @return a {@link CompletableFuture} that completes when the IoT Thing has
been created or if it already exists
     */
    public CompletableFuture<Void> createThingIfNotExistsAsync(String thingName) {
        IotAsyncClient iotClient = IotAsyncClient.builder()
                .region(Region.US_EAST_1)
                .build();

        CreateThingRequest request = CreateThingRequest.builder()
                .thingName(thingName)
                .build();

        return iotClient.createThing(request)
                .whenComplete((response, exception) -> {
                    if (exception != null) {
                        if (exception instanceof ResourceAlreadyExistsException) {
                            logger.info(" IoT Thing already exists: " + thingName);
                        } else {
                            throw new CompletionException("Failed to create IoT
Thing: " + thingName, exception);
                        }
                    } else {
                        logger.info("IoT Thing created: " + response.thingName());
                    }
                })
                .thenApply(response -> null);
    }


    /**
     * Deletes a model manifest.
     *
     * @param name the name of the model manifest to delete
     * @return a {@link CompletableFuture} that completes when the model manifest
has been deleted
     */
    public CompletableFuture<Void> deleteModelManifestAsync(String name) {
        DeleteModelManifestRequest request = DeleteModelManifestRequest.builder()
                .name(name)
                .build();

        return getAsyncClient().deleteModelManifest(request)
```

```
                .handle((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause() != null ?
exception.getCause() : exception;
                        if (cause instanceof ResourceNotFoundException) {
                            throw (ResourceNotFoundException) cause;
                        }
                        throw new RuntimeException("Failed to delete the model
manifest: " + cause);
                    }
                    logger.info("{} was successfully deleted", name);
                    return null;
                });
    }


    /**
     * Deletes a signal catalog.
     *
     * @param name the name of the signal catalog to delete
     * @return a {@link CompletableFuture} that completes when the signal catalog is
deleted
     */
    public CompletableFuture<Void> deleteSignalCatalogAsync(String name) {
        DeleteSignalCatalogRequest request = DeleteSignalCatalogRequest.builder()
                .name(name)
                .build();

        return getAsyncClient().deleteSignalCatalog(request)
                .handle((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause() != null ?
exception.getCause() : exception;
                        if (cause instanceof ResourceNotFoundException) {
                            throw (ResourceNotFoundException) cause;
                        }
                        throw new RuntimeException("Failed to delete the signal
catalog: " + cause);
                    }
                    logger.info("{} was successfully deleted", name);
                    return null;
                });
    }
```

```
   /**
    * Asynchronously retrieves a list of all nodes in the specified signal catalog.
    *
    * @param signalCatalogName the name of the signal catalog to retrieve nodes for
    * @return a {@link CompletableFuture} that, when completed, contains a {@link
List} of {@link Node} objects
    * representing all the nodes in the specified signal catalog
    */
   public CompletableFuture<List<Node>> listSignalCatalogNodeAsync(String
signalCatalogName) {
       ListSignalCatalogNodesRequest request =
ListSignalCatalogNodesRequest.builder()
               .name(signalCatalogName)
               .build();

       List<Node> allNodes = new ArrayList<>();

       return getAsyncClient().listSignalCatalogNodesPaginator(request)
               .subscribe(response -> allNodes.addAll(response.nodes()))
               .thenApply(v -> allNodes);
   }



   /**
    * Creates a model manifest.
    *
    * @param name             the name of the model manifest to create
    * @param signalCatalogArn the Amazon Resource Name (ARN) of the signal catalog
    * @param nodes            a list of nodes to include in the model manifest
    * @return a {@link CompletableFuture} that completes with the ARN of the
created model manifest
    */
   public CompletableFuture<String> createModelManifestAsync(String name,
                                                              String
signalCatalogArn,
                                                              List<Node> nodes) {
       // Extract the fully qualified names (FQNs) from each Node in the provided
list.
       List<String> fqnList = nodes.stream()
               .map(node -> {
                   if (node.sensor() != null) {
                       return node.sensor().fullyQualifiedName();
                   } else if (node.branch() != null) {
```

```
                        return node.branch().fullyQualifiedName();
                    } else if (node.attribute() != null) {
                        return node.attribute().fullyQualifiedName();
                    } else {
                        throw new RuntimeException("Unsupported node type");
                    }
                })
                .toList();

        CreateModelManifestRequest request = CreateModelManifestRequest.builder()
                .name(name)
                .signalCatalogArn(signalCatalogArn)
                .nodes(fqnList)
                .build();


        CompletableFuture<String> result = new CompletableFuture<>();
        getAsyncClient().createModelManifest(request)
                .whenComplete((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause() != null ?
exception.getCause() : exception;

                        if (cause instanceof InvalidSignalsException) {
                            result.completeExceptionally(new
CompletionException("The request contains signals that aren't valid: " +
cause.getMessage(), cause));
                        } else {
                            result.completeExceptionally(new
CompletionException("Failed to create model manifest: " + exception.getMessage(),
exception));
                        }
                    } else {
                        result.complete(response.arn()); // Complete successfully
with the ARN
                    }
                });

        return result;
    }


    /**
     * Deletes a fleet based on the provided fleet ID.
```

```
     *
     * @param fleetId the ID of the fleet to be deleted
     */
    public CompletableFuture<Void> deleteFleetAsync(String fleetId) {
        DeleteFleetRequest request = DeleteFleetRequest.builder()
                .fleetId(fleetId)
                .build();

        return getAsyncClient().deleteFleet(request)
                .handle((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause() != null ?
exception.getCause() : exception;
                        if (cause instanceof ResourceNotFoundException) {
                            throw (ResourceNotFoundException) cause;
                        }
                        throw new RuntimeException("Failed to delete the fleet: " +
cause);
                    }
                    logger.info("{} was successfully deleted", fleetId);
                    return null;
                });
    }



    /**
     * Creates a new fleet.
     *
     * @param catARN  the Amazon Resource Name (ARN) of the signal catalog to
associate with the fleet
     * @param fleetId the unique identifier for the fleet
     * @return a {@link CompletableFuture} that completes with the ID of the created
fleet
     */
    public CompletableFuture<String> createFleetAsync(String catARN, String fleetId)
{
        CreateFleetRequest fleetRequest = CreateFleetRequest.builder()
                .fleetId(fleetId)
                .signalCatalogArn(catARN)
                .description("Built using the AWS For Java V2")
                .build();

        CompletableFuture<String> result = new CompletableFuture<>();
```

```
        getAsyncClient().createFleet(fleetRequest)
                .whenComplete((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause() != null ?
 exception.getCause() : exception;

                        if (cause instanceof ResourceNotFoundException) {
                            result.completeExceptionally(cause);
                        } else {
                            result.completeExceptionally(new RuntimeException("An
 unexpected error occurred", cause));
                        }
                    } else {
                        result.complete(response.id());
                    }
                });

        return result;
    }
}
```

Azioni

`createDecoderManifest`

Il seguente esempio di codice mostra come usare. `createDecoderManifest`

SDK per Java 2.x

> **Note**
>
> C'è 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 a new decoder manifest.
     *
     * @param name             the name of the decoder manifest
     * @param modelManifestArn the ARN of the model manifest
```

```
    * @return a {@link CompletableFuture} that completes with the ARN of the
created decoder manifest
    */
   public CompletableFuture<String> createDecoderManifestAsync(String name, String
modelManifestArn) {
       String interfaceId = "can0";
       NetworkInterface networkInterface = NetworkInterface.builder()
               .interfaceId(interfaceId)
               .type(NetworkInterfaceType.CAN_INTERFACE)
               .canInterface(CanInterface.builder()
                       .name("canInterface0")
                       .protocolName("CAN")
                       .protocolVersion("1.0")
                       .build())
               .build();

       // Vehicle.Powertrain.EngineRPM decoder.
       SignalDecoder engineRpmDecoder = SignalDecoder.builder()
               .fullyQualifiedName("Vehicle.Powertrain.EngineRPM")
               .interfaceId(interfaceId)
               .type(SignalDecoderType.CAN_SIGNAL)
               .canSignal(CanSignal.builder()
                       .messageId(100)
                       .isBigEndian(false)
                       .isSigned(false)
                       .startBit(0)
                       .length(16)
                       .factor(1.0)
                       .offset(0.0)
                       .build())
               .build();

       // Vehicle.Powertrain.VehicleSpeed decoder.
       SignalDecoder vehicleSpeedDecoder = SignalDecoder.builder()
               .fullyQualifiedName("Vehicle.Powertrain.VehicleSpeed")
               .interfaceId(interfaceId)
               .type(SignalDecoderType.CAN_SIGNAL)
               .canSignal(CanSignal.builder()
                       .messageId(101)
                       .isBigEndian(false)
                       .isSigned(false)
                       .startBit(16)
                       .length(16)
                       .factor(1.0)
```

```
                        .offset(0.0)
                        .build())
                .build();

        CreateDecoderManifestRequest request =
CreateDecoderManifestRequest.builder()
                .name(name)
                .modelManifestArn(modelManifestArn)
                .networkInterfaces(List.of(networkInterface))
                .signalDecoders(List.of(engineRpmDecoder, vehicleSpeedDecoder))
                .build();

        CompletableFuture<String> result = new CompletableFuture<>();

        getAsyncClient().createDecoderManifest(request)
                .whenComplete((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause() != null ?
exception.getCause() : exception;

                        if (cause instanceof DecoderManifestValidationException) {
                            result.completeExceptionally(new
CompletionException("The request contains signal decoders with validation errors: "
+ cause.getMessage(), cause));
                        } else {
                            result.completeExceptionally(new
CompletionException("Failed to create decoder manifest: " + exception.getMessage(),
exception));
                        }
                    } else {
                        result.complete(response.arn()); // Complete successfully
with the ARN
                    }
                });

        return result;
    }
```

- Per i dettagli sull'API, createDecoderManifestconsulta AWS SDK for Java 2.x API Reference.

`createFleet`

Il seguente esempio di codice mostra come utilizzare`createFleet`.

SDK per Java 2.x

> **Note**
>
> C'è 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 a new fleet.
    *
    * @param catARN  the Amazon Resource Name (ARN) of the signal catalog to
associate with the fleet
    * @param fleetId the unique identifier for the fleet
    * @return a {@link CompletableFuture} that completes with the ID of the created
fleet
    */
   public CompletableFuture<String> createFleetAsync(String catARN, String fleetId)
{
       CreateFleetRequest fleetRequest = CreateFleetRequest.builder()
               .fleetId(fleetId)
               .signalCatalogArn(catARN)
               .description("Built using the AWS For Java V2")
               .build();

       CompletableFuture<String> result = new CompletableFuture<>();
       getAsyncClient().createFleet(fleetRequest)
               .whenComplete((response, exception) -> {
                   if (exception != null) {
                       Throwable cause = exception.getCause() != null ?
exception.getCause() : exception;

                       if (cause instanceof ResourceNotFoundException) {
                           result.completeExceptionally(cause);
                       } else {
                           result.completeExceptionally(new RuntimeException("An
unexpected error occurred", cause));
                       }
```

```
                    } else {
                        result.complete(response.id());
                    }
                });

        return result;
    }
```

- Per i dettagli sull'API, consulta CreateFleet in AWS SDK for Java 2.x API Reference.

`createModelManifest`

Il seguente esempio di codice mostra come utilizzare. `createModelManifest`

SDK per Java 2.x

> **Note**
>
> C'è 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 a model manifest.
     *
     * @param name             the name of the model manifest to create
     * @param signalCatalogArn the Amazon Resource Name (ARN) of the signal catalog
     * @param nodes            a list of nodes to include in the model manifest
     * @return a {@link CompletableFuture} that completes with the ARN of the
created model manifest
     */
    public CompletableFuture<String> createModelManifestAsync(String name,
                                                               String
signalCatalogArn,
                                                               List<Node> nodes) {
        // Extract the fully qualified names (FQNs) from each Node in the provided
list.
        List<String> fqnList = nodes.stream()
                .map(node -> {
                    if (node.sensor() != null) {
```

```
                        return node.sensor().fullyQualifiedName();
                    } else if (node.branch() != null) {
                        return node.branch().fullyQualifiedName();
                    } else if (node.attribute() != null) {
                        return node.attribute().fullyQualifiedName();
                    } else {
                        throw new RuntimeException("Unsupported node type");
                    }
                })
                .toList();

        CreateModelManifestRequest request = CreateModelManifestRequest.builder()
                .name(name)
                .signalCatalogArn(signalCatalogArn)
                .nodes(fqnList)
                .build();


        CompletableFuture<String> result = new CompletableFuture<>();
        getAsyncClient().createModelManifest(request)
                .whenComplete((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause() != null ?
exception.getCause() : exception;

                        if (cause instanceof InvalidSignalsException) {
                            result.completeExceptionally(new
CompletionException("The request contains signals that aren't valid: " +
cause.getMessage(), cause));
                        } else {
                            result.completeExceptionally(new
CompletionException("Failed to create model manifest: " + exception.getMessage(),
exception));
                        }
                    } else {
                        result.complete(response.arn()); // Complete successfully
with the ARN
                    }
                });

        return result;
    }
```

- Per i dettagli sull'API, createModelManifestconsulta AWS SDK for Java 2.x API Reference.

`createSignalCatalog`

Il seguente esempio di codice mostra come utilizzare`createSignalCatalog`.

SDK per Java 2.x

> **Note**
>
> C'è 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 a signal catalog.
    *
    * @param signalCatalogName the name of the signal catalog to be created
    * @return a {@link CompletableFuture} that completes with the Amazon Resource
Name (ARN) of the created signal catalog
    */
   public CompletableFuture<String> createSignalCatalogAsync(String
signalCatalogName) {
       return deleteSignalCatalogIfExistsAsync(signalCatalogName)
               .thenCompose(ignored -> delayAsync(2000)) // Wait for 2 seconds
               .thenCompose(ignored -> {
                   List<Node> nodes = List.of(
                           Node.builder().branch(
                                   Branch.builder()
                                           .fullyQualifiedName("Vehicle")
                                           .description("Root branch")
                                           .build()
                           ).build(),
                           Node.builder().branch(
                                   Branch.builder()

.fullyQualifiedName("Vehicle.Powertrain")
                                           .description("Powertrain branch")
                                           .build()
                           ).build(),
                           Node.builder().sensor(
```

```
                                    Sensor.builder()

.fullyQualifiedName("Vehicle.Powertrain.EngineRPM")
                                            .description("Engine RPM")
                                            .dataType(NodeDataType.DOUBLE)
                                            .unit("rpm")
                                            .build()
                                ).build(),
                                Node.builder().sensor(
                                        Sensor.builder()

.fullyQualifiedName("Vehicle.Powertrain.VehicleSpeed")
                                            .description("Vehicle Speed")
                                            .dataType(NodeDataType.DOUBLE)
                                            .unit("km/h")
                                            .build()
                                ).build()
                        );

                        CreateSignalCatalogRequest request =
CreateSignalCatalogRequest.builder()
                                .name(signalCatalogName)
                                .nodes(nodes)
                                .build();

                        CompletableFuture<String> result = new CompletableFuture<>();

                        getAsyncClient().createSignalCatalog(request)
                                .whenComplete((response, exception) -> {
                                    if (exception != null) {
                                        Throwable cause = exception.getCause() != null ?
exception.getCause() : exception;

                                        if (cause instanceof ValidationException) {
                                            result.completeExceptionally(cause);
                                        } else {
                                            result.completeExceptionally(new
RuntimeException("Error creating the catalog", cause));
                                        }
                                    } else {
                                        result.complete(response.arn());
                                    }
                                });
```

```
                    return result;
                });
    }
```

- Per i dettagli sull'API, createSignalCatalogconsulta AWS SDK for Java 2.x API Reference.

`createVehicle`

Il seguente esempio di codice mostra come utilizzare`createVehicle`.

SDK per Java 2.x

> **Note**
>
> C'è 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 a new vehicle in the system.
     *
     * @param vecName     the name of the vehicle to be created
     * @param manifestArn the Amazon Resource Name (ARN) of the model manifest for
the vehicle
     * @param decArn      the Amazon Resource Name (ARN) of the decoder manifest for
the vehicle
     * @return a {@link CompletableFuture} that completes when the vehicle has been
created, or throws a
     */
    public CompletableFuture<Void> createVehicleAsync(String vecName, String
manifestArn, String decArn) {
        CreateVehicleRequest request = CreateVehicleRequest.builder()
                .vehicleName(vecName)
                .modelManifestArn(manifestArn)
                .decoderManifestArn(decArn)
                .build();

        CompletableFuture<Void> result = new CompletableFuture<>();
        getAsyncClient().createVehicle(request)
```

```
                .whenComplete((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception instanceof CompletionException ?
exception.getCause() : exception;

                        if (cause instanceof ResourceNotFoundException) {
                            result.completeExceptionally(cause);
                        } else {
                            result.completeExceptionally(new
RuntimeException("Failed to create vehicle: " + cause.getMessage(), cause));
                        }
                    } else {
                        logger.info("Vehicle '{}' created successfully.", vecName);
                        result.complete(null); // mark future as complete
                    }
                });

        return result;
    }
```

- Per i dettagli sull'API, consulta CreateVehicle in API Reference.AWS SDK for Java 2.x

`deleteDecoderManifest`

Il seguente esempio di codice mostra come utilizzare. `deleteDecoderManifest`

SDK per Java 2.x

> **Note**
>
> C'è 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 decoder manifest.
     *
     * @param name the name of the decoder manifest to delete
     * @return a {@link CompletableFuture} that completes when the decoder manifest
has been deleted
```

```
     */
    public CompletableFuture<Void> deleteDecoderManifestAsync(String name) {
        return
getAsyncClient().deleteDecoderManifest(DeleteDecoderManifestRequest.builder().name(name).bu
                .handle((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause() != null ?
exception.getCause() : exception;
                        if (cause instanceof ResourceNotFoundException) {
                            throw (ResourceNotFoundException) cause;
                        }
                        throw new RuntimeException("Failed to delete the decoder
manifest: " + cause);
                    }
                    return null;
                });
    }
```

- Per i dettagli sull'API, deleteDecoderManifestconsulta AWS SDK for Java 2.x API Reference.

`deleteFleet`

Il seguente esempio di codice mostra come utilizzare`deleteFleet`.

SDK per Java 2.x

> **Note**
>
> C'è 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 fleet based on the provided fleet ID.
     *
     * @param fleetId the ID of the fleet to be deleted
     */
    public CompletableFuture<Void> deleteFleetAsync(String fleetId) {
        DeleteFleetRequest request = DeleteFleetRequest.builder()
                .fleetId(fleetId)
```

```
                .build();

        return getAsyncClient().deleteFleet(request)
                .handle((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause() != null ?
exception.getCause() : exception;
                        if (cause instanceof ResourceNotFoundException) {
                            throw (ResourceNotFoundException) cause;
                        }
                        throw new RuntimeException("Failed to delete the fleet: " +
cause);
                    }
                    logger.info("{} was successfully deleted", fleetId);
                    return null;
                });
    }
```

- Per i dettagli sull'API, consulta DeleteFleet in AWS SDK for Java 2.x API Reference.

`deleteModelManifest`

Il seguente esempio di codice mostra come utilizzare. `deleteModelManifest`

SDK per Java 2.x

> **Note**
>
> C'è 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 model manifest.
     *
     * @param name the name of the model manifest to delete
     * @return a {@link CompletableFuture} that completes when the model manifest
has been deleted
     */
    public CompletableFuture<Void> deleteModelManifestAsync(String name) {
```

```
        DeleteModelManifestRequest request = DeleteModelManifestRequest.builder()
                .name(name)
                .build();

        return getAsyncClient().deleteModelManifest(request)
                .handle((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause() != null ?
exception.getCause() : exception;
                        if (cause instanceof ResourceNotFoundException) {
                            throw (ResourceNotFoundException) cause;
                        }
                        throw new RuntimeException("Failed to delete the model
manifest: " + cause);
                    }
                    logger.info("{} was successfully deleted", name);
                    return null;
                });
    }
```

- Per i dettagli sull'API, deleteModelManifestconsulta AWS SDK for Java 2.x API Reference.

`deleteSignalCatalog`

Il seguente esempio di codice mostra come utilizzare`deleteSignalCatalog`.

SDK per Java 2.x

> **Note**
>
> C'è 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 signal catalog.
     *
     * @param name the name of the signal catalog to delete
     * @return a {@link CompletableFuture} that completes when the signal catalog is
deleted
```

```
     */
    public CompletableFuture<Void> deleteSignalCatalogAsync(String name) {
        DeleteSignalCatalogRequest request = DeleteSignalCatalogRequest.builder()
                .name(name)
                .build();

        return getAsyncClient().deleteSignalCatalog(request)
                .handle((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause() != null ?
exception.getCause() : exception;
                        if (cause instanceof ResourceNotFoundException) {
                            throw (ResourceNotFoundException) cause;
                        }
                        throw new RuntimeException("Failed to delete the signal
catalog: " + cause);
                    }
                    logger.info("{} was successfully deleted", name);
                    return null;
                });
    }
```

- Per i dettagli sull'API, deleteSignalCatalogconsulta AWS SDK for Java 2.x API Reference.

`deleteVehicle`

Il seguente esempio di codice mostra come utilizzare`deleteVehicle`.

SDK per Java 2.x

> **Note**
>
> C'è 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 vehicle with the specified name.
     *
     * @param vecName the name of the vehicle to be deleted
```

```
     * @return a {@link CompletableFuture} that completes when the vehicle has been
deleted
     */
    public CompletableFuture<Void> deleteVehicleAsync(String vecName) {
        DeleteVehicleRequest request = DeleteVehicleRequest.builder()
                .vehicleName(vecName)
                .build();

        return getAsyncClient().deleteVehicle(request)
                .handle((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause() != null ?
exception.getCause() : exception;
                        if (cause instanceof ResourceNotFoundException) {
                            throw (ResourceNotFoundException) cause;
                        }
                        throw new RuntimeException("Failed to delete the vehicle: "
+ cause);
                    }
                    return null;
                });
    }
```

- Per i dettagli sull'API, consulta DeleteVehicle in API Reference.AWS SDK for Java 2.x

`getDecoderManifest`

Il seguente esempio di codice mostra come utilizzare. `getDecoderManifest`

SDK per Java 2.x

> **Note**
>
> C'è 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 for the decoder manifest to become active.
     *
     * @param decoderName the name of the decoder to wait for
```

```
    * @return a {@link CompletableFuture} that completes when the decoder manifest
becomes active, or exceptionally if an error occurs or the manifest becomes invalid
    */
   public CompletableFuture<Void> waitForDecoderManifestActiveAsync(String
decoderName) {
       CompletableFuture<Void> result = new CompletableFuture<>();

       ScheduledExecutorService scheduler =
Executors.newSingleThreadScheduledExecutor();
       AtomicInteger secondsElapsed = new AtomicInteger(0);
       AtomicReference<ManifestStatus> lastStatus = new
AtomicReference<>(ManifestStatus.DRAFT);

       logger.info(" Elapsed: 0s | Decoder Status: DRAFT");

       final Runnable pollTask = new Runnable() {
           @Override
           public void run() {
               int elapsed = secondsElapsed.incrementAndGet();

               // Check status every 5 seconds
               if (elapsed % 5 == 0) {
                   GetDecoderManifestRequest request =
GetDecoderManifestRequest.builder()
                           .name(decoderName)
                           .build();

                   getAsyncClient().getDecoderManifest(request)
                           .whenComplete((response, exception) -> {
                               if (exception != null) {
                                   Throwable cause = exception instanceof
CompletionException ? exception.getCause() : exception;

                                   scheduler.shutdown();
                                   if (cause instanceof ResourceNotFoundException)
{
                                       result.completeExceptionally(new
RuntimeException("Decoder manifest not found: " + cause.getMessage(), cause));
                                   } else {
                                       result.completeExceptionally(new
RuntimeException("Error while polling decoder manifest status: " +
exception.getMessage(), exception));
                                   }
                                   return;
```

```
                                }

                                ManifestStatus status = response.status();
                                lastStatus.set(status);

                                if (status == ManifestStatus.ACTIVE) {
                                    logger.info("\r Elapsed: {}s | Decoder Status:
ACTIVE", elapsed);
                                    scheduler.shutdown();
                                    result.complete(null);
                                } else if (status == ManifestStatus.INVALID) {
                                    logger.info("\r Elapsed: {}s | Decoder Status:
INVALID", elapsed);
                                    scheduler.shutdown();
                                    result.completeExceptionally(new
RuntimeException("Decoder manifest became INVALID. Cannot proceed."));
                                } else {
                                    logger.info("\r# Elapsed: {}s | Decoder Status:
{}", elapsed, status);
                                }
                            });
                } else {
                    logger.info("\r Elapsed: {}s | Decoder Status: {}", elapsed,
lastStatus.get());
                }
            }
        };

        // Start the task with an initial delay of 1 second, and repeat every second
        scheduler.scheduleAtFixedRate(pollTask, 1, 1, TimeUnit.SECONDS);
        return result;
    }
```

- Per i dettagli sull'API, [getDecoderManifest](consulta AWS SDK for Java 2.x API Reference.

`getModelManifest`

Il seguente esempio di codice mostra come utilizzare`getModelManifest`.

SDK per Java 2.x

> **Note**
>
> C'è 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 for the specified model manifest to become active.
    *
    * @param manifestName the name of the model manifest to wait for
    */
   public CompletableFuture<Void> waitForModelManifestActiveAsync(String
manifestName) {
       CompletableFuture<Void> result = new CompletableFuture<>();

       ScheduledExecutorService scheduler =
Executors.newSingleThreadScheduledExecutor();
       AtomicInteger secondsElapsed = new AtomicInteger(0);
       AtomicReference<ManifestStatus> lastStatus = new
AtomicReference<>(ManifestStatus.DRAFT);

       logger.info("Elapsed: 0s | Status: DRAFT");

       final Runnable pollTask = new Runnable() {
           @Override
           public void run() {
               int elapsed = secondsElapsed.incrementAndGet();

               // Only check status every 5 seconds
               if (elapsed % 5 == 0) {
                   GetModelManifestRequest request =
GetModelManifestRequest.builder()
                           .name(manifestName)
                           .build();

                   getAsyncClient().getModelManifest(request)
                           .whenComplete((response, exception) -> {
                               if (exception != null) {
```

```
                                    Throwable cause = exception instanceof
CompletionException ? exception.getCause() : exception;

                                    scheduler.shutdown();
                                    if (cause instanceof ResourceNotFoundException)
{
                                        result.completeExceptionally(new
RuntimeException("Model manifest not found: " + cause.getMessage(), cause));
                                    } else {
                                        result.completeExceptionally(new
RuntimeException("Error while polling model manifest status: " +
exception.getMessage(), exception));
                                    }
                                    return;
                                }

                                ManifestStatus status = response.status();
                                lastStatus.set(status);

                                if (status == ManifestStatus.ACTIVE) {
                                    logger.info("\rElapsed: {}s | Status: ACTIVE",
elapsed);
                                    scheduler.shutdown();
                                    result.complete(null);
                                } else if (status == ManifestStatus.INVALID) {
                                    logger.info("\rElapsed: {}s | Status: INVALID",
elapsed);
                                    scheduler.shutdown();
                                    result.completeExceptionally(new
RuntimeException("Model manifest became INVALID. Cannot proceed."));
                                } else {
                                    logger.info("\rElapsed: {}s | Status: {}",
elapsed, status);
                                }
                            });
                } else {
                    logger.info("\rElapsed: {}s | Status: {}", elapsed,
lastStatus.get());
                }
            }
        };

        // Start the task with an initial delay of 1 second, and repeat every second
        scheduler.scheduleAtFixedRate(pollTask, 1, 1, TimeUnit.SECONDS);
```

```
        return result;
    }
```

- Per i dettagli sull'API, getModelManifestconsulta AWS SDK for Java 2.x API Reference.

`getVehicle`

Il seguente esempio di codice mostra come utilizzare`getVehicle`.

SDK per Java 2.x

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
    /**
     * Fetches the details of a vehicle.
     *
     * @param vehicleName the name of the vehicle to fetch details for
     * @return a {@link CompletableFuture} that completes when the vehicle details
have been fetched
     */
    public CompletableFuture<Void> getVehicleDetailsAsync(String vehicleName) {
        GetVehicleRequest request = GetVehicleRequest.builder()
                .vehicleName(vehicleName)
                .build();

        CompletableFuture<Void> result = new CompletableFuture<>();

        getAsyncClient().getVehicle(request)
                .whenComplete((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception instanceof CompletionException ?
exception.getCause() : exception;

                        if (cause instanceof ResourceNotFoundException) {
```

```
                            result.completeExceptionally(cause); // don't rewrap
                        } else {
                            result.completeExceptionally(new
RuntimeException("Failed to fetch vehicle details: " + cause.getMessage(), cause));
                        }
                    } else {
                        Map<String, Object> details = new HashMap<>();
                        details.put("vehicleName", response.vehicleName());
                        details.put("arn", response.arn());
                        details.put("modelManifestArn",
response.modelManifestArn());
                        details.put("decoderManifestArn",
response.decoderManifestArn());
                        details.put("attributes", response.attributes());
                        details.put("creationTime",
response.creationTime().toString());
                        details.put("lastModificationTime",
response.lastModificationTime().toString());

                        logger.info("Vehicle Details:");
                        details.forEach((key, value) -> logger.info("• {} : {}",
key, value));

                        result.complete(null); // mark as successful
                    }
                });

        return result;
    }
```

- Per i dettagli sull'API, consulta GetVehicle in AWS SDK for Java 2.x API Reference.

`listSignalCatalogNodes`

Il seguente esempio di codice mostra come utilizzare`listSignalCatalogNodes`.

SDK per Java 2.x

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
   /**
    * Asynchronously retrieves a list of all nodes in the specified signal catalog.
    *
    * @param signalCatalogName the name of the signal catalog to retrieve nodes for
    * @return a {@link CompletableFuture} that, when completed, contains a {@link
List} of {@link Node} objects
    * representing all the nodes in the specified signal catalog
    */
   public CompletableFuture<List<Node>> listSignalCatalogNodeAsync(String
signalCatalogName) {
       ListSignalCatalogNodesRequest request =
ListSignalCatalogNodesRequest.builder()
               .name(signalCatalogName)
               .build();

       List<Node> allNodes = new ArrayList<>();

       return getAsyncClient().listSignalCatalogNodesPaginator(request)
               .subscribe(response -> allNodes.addAll(response.nodes()))
               .thenApply(v -> allNodes);
   }
```

- Per i dettagli sull'API, consulta listSignalCatalogNodes in AWS SDK for Java 2.x API Reference.

`updateDecoderManifest`

Il seguente esempio di codice mostra come utilizzare`updateDecoderManifest`.

SDK per Java 2.x

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
   /**
    * Updates the decoder manifest with the given name.
    *
    * @param name the name of the decoder manifest to update
    * @return a {@link CompletableFuture} that completes when the update operation
is finished
    */
   public CompletableFuture<Void> updateDecoderManifestAsync(String name) {
       UpdateDecoderManifestRequest request =
UpdateDecoderManifestRequest.builder()
               .name(name)
               .status(ManifestStatus.ACTIVE)
               .build();

       return getAsyncClient().updateDecoderManifest(request)
               .whenComplete((response, exception) -> {
                   if (exception != null) {
                       throw new CompletionException("Failed to update decoder
manifest: " + exception.getMessage(), exception);
                   }
               })
               .thenApply(response -> null);
   }
```

- Per i dettagli sull'API, updateDecoderManifestconsulta AWS SDK for Java 2.x API Reference.

`updateModelManifest`

Il seguente esempio di codice mostra come utilizzare`updateModelManifest`.

SDK per Java 2.x

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
    /**
     * Updates the model manifest.
     *
     * @param name the name of the model manifest to update
     */
    public void updateModelManifestAsync(String name) {
        UpdateModelManifestRequest request = UpdateModelManifestRequest.builder()
                .name(name)
                .status(ManifestStatus.ACTIVE)
                .build();

        getAsyncClient().updateModelManifest(request)
                .whenComplete((response, exception) -> {
                    if (exception != null) {
                        throw new CompletionException("Failed to update model
manifest: " + exception.getMessage(), exception);
                    }
                })
                .thenApply(response -> null);
    }
```

- Per i dettagli sull'API, updateModelManifestconsulta AWS SDK for Java 2.x API Reference.

AWS IoT SiteWise 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 SiteWise.

Le nozioni di base sono esempi di codice che mostrano come eseguire le operazioni essenziali all'interno di un servizio.

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.

Ogni esempio include un collegamento al codice sorgente completo, in cui è possibile trovare istruzioni su come configurare ed eseguire il codice nel contesto.

Nozioni di base

Salve AWS IoT SiteWise

L'esempio di codice seguente mostra come iniziare a utilizzare AWS IoT SiteWise.

SDK per Java 2.x

> **Note**
>
> C'è 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 HelloSitewise {
    private static final Logger logger =
 LoggerFactory.getLogger(HelloSitewise.class);
    public static void main(String[] args) {
         fetchAssetModels();
    }

    /**
     * Fetches asset models using the provided {@link IoTSiteWiseAsyncClient}.
     */
    public static void fetchAssetModels() {
        IoTSiteWiseAsyncClient siteWiseAsyncClient =
 IoTSiteWiseAsyncClient.create();
        ListAssetModelsRequest assetModelsRequest = ListAssetModelsRequest.builder()
            .assetModelTypes(AssetModelType.ASSET_MODEL)
            .build();

        // Asynchronous paginator - process paginated results.
        ListAssetModelsPublisher listModelsPaginator =
 siteWiseAsyncClient.listAssetModelsPaginator(assetModelsRequest);
        CompletableFuture<Void> future = listModelsPaginator.subscribe(response -> {
```

```
            response.assetModelSummaries().forEach(assetSummary ->
                logger.info("Asset Model Name: {} ", assetSummary.name())
            );
        });

        // Wait for the asynchronous operation to complete
        future.join();
    }
}
```

- Per i dettagli sull'API, ListAssetModelsconsulta AWS SDK for Java 2.x API Reference.

Argomenti

- Nozioni di base
- Azioni

Nozioni di base

Informazioni di base

L'esempio di codice seguente mostra come:

- Crea un modello di AWS IoT SiteWise asset.
- Crea una AWS IoT SiteWise risorsa.
- Recupera i valori dell'ID della proprietà.
- Invia dati a una AWS IoT SiteWise risorsa.
- Recupera il valore della proprietà AWS IoT SiteWise Asset.
- Crea un AWS IoT SiteWise portale.
- Crea un AWS IoT SiteWise gateway.
- Descrivi il AWS IoT SiteWise Gateway.
- Eliminare gli AWS IoT SiteWise asset.

SDK per Java 2.x

> **Note**
>
> C'è altro da fare 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 AWS IoT SiteWise funzionalità.

```
public class SitewiseScenario {

    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    private static final Logger logger =
 LoggerFactory.getLogger(SitewiseScenario.class);
    static Scanner scanner = new Scanner(System.in);

    private static final String ROLES_STACK = "RoleSitewise";

    static SitewiseActions sitewiseActions = new SitewiseActions();

    public static void main(String[] args) throws Throwable {
        Scanner scanner = new Scanner(System.in);
        String contactEmail = "user@mydomain.com"; // Change email address.
        String assetModelName = "MyAssetModel1";
        String assetName = "MyAsset1" ;
        String portalName = "MyPortal1" ;
        String gatewayName = "MyGateway1" ;
        String myThing =  "MyThing1" ;

        logger.info("""
            AWS IoT SiteWise is a fully managed software-as-a-service (SaaS) that
            makes it easy to collect, store, organize, and monitor data from
 industrial equipment and processes.
            It is designed to help industrial and manufacturing organizations
 collect data from their equipment and
            processes, and use that data to make informed decisions about their
 operations.

            One of the key features of AWS IoT SiteWise is its ability to connect to
 a wide range of industrial
```

```
            equipment and systems, including programmable logic controllers (PLCs),
sensors, and other
            industrial devices. It can collect data from these devices and organize
it into a unified data model,
            making it easier to analyze and gain insights from the data. AWS IoT
SiteWise also provides tools for
            visualizing the data, setting up alarms and alerts, and generating
reports.

            Another key feature of AWS IoT SiteWise is its ability to scale to
handle large volumes of data.
            It can collect and store data from thousands of devices and process
millions of data points per second,
            making it suitable for large-scale industrial operations. Additionally,
AWS IoT SiteWise is designed
            to be secure and compliant, with features like role-based access
controls, data encryption,
            and integration with other AWS services for additional security and
compliance features.

            Let's get started...
            """);

        waitForInputToContinue(scanner);
        logger.info(DASHES);

        try {
            runScenario(assetModelName, assetName, portalName, contactEmail,
gatewayName, myThing);
        } catch (RuntimeException e) {
           logger.info(e.getMessage());
        }
    }

    public static void runScenario(String assetModelName, String assetName,  String
portalName, String contactEmail, String gatewayName, String myThing) throws
Throwable {
        logger.info("Use AWS CloudFormation to create an IAM role that is required
for this scenario.");
        CloudFormationHelper.deployCloudFormationStack(ROLES_STACK);
        Map<String, String> stackOutputs =
CloudFormationHelper.getStackOutputsAsync(ROLES_STACK).join();
        String iamRole = stackOutputs.get("SitewiseRoleArn");
        logger.info("The ARN of the IAM role is {}",iamRole);
```

```
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("1. Create an AWS SiteWise Asset Model");
        logger.info("""
             An AWS IoT SiteWise Asset Model is a way to represent the physical
assets, such as equipment,
             processes, and systems, that exist in an industrial environment. This
model provides a structured and
             hierarchical representation of these assets, allowing users to define
the relationships and properties
             of each asset.

             This scenario creates two asset model properties: temperature and
humidity.
            """);
        waitForInputToContinue(scanner);
        String assetModelId = null;
        try {
            CreateAssetModelResponse response =
sitewiseActions.createAssetModelAsync(assetModelName).join();
            assetModelId = response.assetModelId();
            logger.info("Asset Model successfully created. Asset Model ID: {}. ",
assetModelId);
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ResourceAlreadyExistsException) {
                try {
                    assetModelId =
sitewiseActions.getAssetModelIdAsync(assetModelName).join();
                    logger.info("The Asset Model {} already exists. The id of the
existing model is {}. Moving on...", assetModelName, assetModelId);
                } catch (CompletionException cex) {
                    logger.error("Exception thrown acquiring the asset model id:
{}", cex.getCause().getCause(), cex);
                    return;
                }
            } else {
                logger.info("An unexpected error occurred: " + cause.getMessage(),
cause);
                return;
            }
        }
        waitForInputToContinue(scanner);
```

```
        logger.info(DASHES);
        logger.info("2. Create an AWS IoT SiteWise Asset");
        logger.info("""
             The IoT SiteWise model that we just created defines the structure and
metadata for your physical assets.
             Now we create an asset from the asset model.

            """);
        logger.info("Let's wait 30 seconds for the asset to be ready.");
        countdown(30);
        waitForInputToContinue(scanner);
        String assetId;
        try {
            CreateAssetResponse response =
sitewiseActions.createAssetAsync(assetName, assetModelId).join();
            assetId = response.assetId();
            logger.info("Asset created with ID: {}", assetId);
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ResourceNotFoundException) {
                logger.info("The asset model id was not found: {}",
cause.getMessage(), cause);
            } else {
                logger.info("An unexpected error occurred: {}", cause.getMessage(),
cause);
            }
            return;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("3. Retrieve the property ID values");
        logger.info("""
             To send data to an asset, we need to get the property ID values. In
this scenario, we access the
             temperature and humidity property ID values.
            """);
        waitForInputToContinue(scanner);
        Map<String, String>  propertyIds = null;
        try {
            propertyIds = sitewiseActions.getPropertyIds(assetModelId).join();
        } catch (CompletionException ce) {
```

```
            Throwable cause = ce.getCause();
            if (cause instanceof IoTSiteWiseException) {
                logger.error("IoTSiteWiseException occurred: {}",
cause.getMessage(), ce);
            } else {
                logger.error("An unexpected error occurred: {}", cause.getMessage(),
ce);
            }
            return;
        }
        String humPropId =  propertyIds.get("Humidity");
        logger.info("The Humidity property Id is {}", humPropId);
        String tempPropId = propertyIds.get("Temperature");
        logger.info("The Temperature property Id is {}", tempPropId);

        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("4. Send data to an AWS IoT SiteWise Asset");
        logger.info("""
            By sending data to an IoT SiteWise Asset, you can aggregate data from
            multiple sources, normalize the data into a standard format, and store
it in a
            centralized location. This makes it easier to analyze and gain insights
from the data.

            In this example, we generate sample temperature and humidity data and
send it to the AWS IoT SiteWise asset.

            """);
        waitForInputToContinue(scanner);
        try {
            sitewiseActions.sendDataToSiteWiseAsync(assetId, tempPropId,
humPropId).join();
            logger.info("Data sent successfully.");
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ResourceNotFoundException) {
                logger.error("The AWS resource was not found: {}",
cause.getMessage(), cause);
            } else {
                logger.error("An unexpected error occurred: {}", cause.getMessage(),
cause);
```

```
            }
            return;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("5. Retrieve the value of the IoT SiteWise Asset property");
        logger.info("""
            IoT SiteWise is an AWS service that allows you to collect, process, and
analyze industrial data
            from connected equipment and sensors. One of the key benefits of reading
an IoT SiteWise property
            is the ability to gain valuable insights from your industrial data.

            """);
        waitForInputToContinue(scanner);
        try {
            Double assetVal = sitewiseActions.getAssetPropValueAsync(tempPropId,
assetId).join();
            logger.info("The property name is: {}", "Temperature");
            logger.info("The value of this property is: {}", assetVal);

            waitForInputToContinue(scanner);

            assetVal = sitewiseActions.getAssetPropValueAsync(humPropId,
assetId).join();
            logger.info("The property name is: {}", "Humidity");
            logger.info("The value of this property is: {}", assetVal);
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
                if (cause instanceof ResourceNotFoundException) {
                    logger.info("The AWS resource was not found: {}",
cause.getMessage(), cause);
                } else {
                    logger.info("An unexpected error occurred: {}",
cause.getMessage(), cause);
                }
                return;
            }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
```

```
        logger.info("6. Create an IoT SiteWise Portal");
        logger.info("""
             An IoT SiteWise Portal allows you to aggregate data from multiple
industrial sources,
             such as sensors, equipment, and control systems, into a centralized
platform.
            """);
        waitForInputToContinue(scanner);
        String portalId;
        try {
            portalId = sitewiseActions.createPortalAsync(portalName, iamRole,
contactEmail).join();
            logger.info("Portal created successfully. Portal ID {}", portalId);
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof IoTSiteWiseException siteWiseEx) {
                logger.error("IoT SiteWise error occurred: Error message: {}, Error
code {}",
                        siteWiseEx.getMessage(),
siteWiseEx.awsErrorDetails().errorCode(), siteWiseEx);
            } else {
                logger.error("An unexpected error occurred: {}",
cause.getMessage());
            }
            return;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("7. Describe the Portal");
        logger.info("""
             In this step, we get a description of the portal and display the portal
URL.
            """);
        waitForInputToContinue(scanner);
        try {
            String portalUrl = sitewiseActions.describePortalAsync(portalId).join();
            logger.info("Portal URL: {}", portalUrl);
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ResourceNotFoundException notFoundException) {
                logger.error("A ResourceNotFoundException occurred: Error message:
{}, Error code {}",
```

```
                        notFoundException.getMessage(),
notFoundException.awsErrorDetails().errorCode(), notFoundException);
            } else {
                logger.error("An unexpected error occurred: {}",
cause.getMessage());
            }
            return;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("8. Create an IoT SiteWise Gateway");
        logger.info(
            """
                IoT SiteWise Gateway serves as the bridge between industrial
equipment, sensors, and the
                cloud-based IoT SiteWise service. It is responsible for securely
collecting, processing, and
                transmitting data from various industrial assets to the IoT SiteWise
platform,
                enabling real-time monitoring, analysis, and optimization of
industrial operations.

                """);
        waitForInputToContinue(scanner);
        String gatewayId = "";
        try {
            gatewayId = sitewiseActions.createGatewayAsync(gatewayName,
myThing).join();
            logger.info("Gateway creation completed successfully. id is {}",
gatewayId );
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof IoTSiteWiseException siteWiseEx) {
                logger.error("IoT SiteWise error occurred: Error message: {}, Error
code {}",
                        siteWiseEx.getMessage(),
siteWiseEx.awsErrorDetails().errorCode(), siteWiseEx);
            } else {
                logger.error("An unexpected error occurred: {}",
cause.getMessage());
            }
            return;
```

```
        }
        logger.info(DASHES);
        logger.info(DASHES);

        logger.info("9. Describe the IoT SiteWise Gateway");
         waitForInputToContinue(scanner);
        try {
            sitewiseActions.describeGatewayAsync(gatewayId)
                .thenAccept(response -> {
                    logger.info("Gateway Name: {}", response.gatewayName());
                    logger.info("Gateway ARN: {}", response.gatewayArn());
                    logger.info("Gateway Platform: {}", response.gatewayPlatform());
                    logger.info("Gateway Creation Date: {}",
response.creationDate());
                }).join();
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ResourceNotFoundException notFoundException) {
                logger.error("A ResourceNotFoundException occurred: Error message:
{}, Error code {}",
                        notFoundException.getMessage(),
notFoundException.awsErrorDetails().errorCode(), notFoundException);
            } else {
                logger.error("An unexpected error occurred: {}", cause.getMessage(),
cause);
            }
            return;
        }
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("10. Delete the AWS IoT SiteWise Assets");
        logger.info(
            """
            Before you can delete the Asset Model, you must delete the assets.

            """);
        logger.info("Would you like to delete the IoT SiteWise Assets? (y/n)");
        String delAns = scanner.nextLine().trim();
        if (delAns.equalsIgnoreCase("y")) {
            logger.info("You selected to delete the SiteWise assets.");
            waitForInputToContinue(scanner);
            try {
                sitewiseActions.deletePortalAsync(portalId).join();
```

```
                logger.info("Portal {} was deleted successfully.", portalId);

            } catch (CompletionException ce) {
                Throwable cause = ce.getCause();
                if (cause instanceof ResourceNotFoundException notFoundException) {
                    logger.error("A ResourceNotFoundException occurred: Error
message: {}, Error code {}",
                            notFoundException.getMessage(),
notFoundException.awsErrorDetails().errorCode(), notFoundException);
                } else {
                    logger.error("An unexpected error occurred: {}",
cause.getMessage());
                }
            }

            try {
                sitewiseActions.deleteGatewayAsync(gatewayId).join();
                logger.info("Gateway {} was deleted successfully.", gatewayId);
            } catch (CompletionException ce) {
                Throwable cause = ce.getCause();
                if (cause instanceof ResourceNotFoundException notFoundException) {
                    logger.error("A ResourceNotFoundException occurred: Error
message: {}, Error code {}",
                            notFoundException.getMessage(),
notFoundException.awsErrorDetails().errorCode(), notFoundException);
                } else {
                    logger.error("An unexpected error occurred: {}",
cause.getMessage());
                }
            }

            try {
                sitewiseActions.deleteAssetAsync(assetId).join();
                logger.info("Request to delete asset {} sent successfully",
assetId);
            } catch (CompletionException ce) {
                Throwable cause = ce.getCause();
                if (cause instanceof ResourceNotFoundException notFoundException) {
                    logger.error("A ResourceNotFoundException occurred: Error
message: {}, Error code {}",
                            notFoundException.getMessage(),
notFoundException.awsErrorDetails().errorCode(), notFoundException);
                } else {
```

```
                    logger.error("An unexpected error occurred: {}",
cause.getMessage());
                }
            }
            logger.info("Let's wait 1 minute for the asset to be deleted.");
            countdown(60);
            waitForInputToContinue(scanner);
            logger.info("Delete the AWS IoT SiteWise Asset Model");
            try {
                sitewiseActions.deleteAssetModelAsync(assetModelId).join();
                logger.info("Asset model deleted successfully.");
            } catch (CompletionException ce) {
                Throwable cause = ce.getCause();
                if (cause instanceof ResourceNotFoundException notFoundException) {
                    logger.error("A ResourceNotFoundException occurred: Error
message: {}, Error code {}",
                            notFoundException.getMessage(),
notFoundException.awsErrorDetails().errorCode(), notFoundException);
                } else {
                    logger.error("An unexpected error occurred: {}",
cause.getMessage());
                }
            }
            waitForInputToContinue(scanner);

        } else {
            logger.info("The resources will not be deleted.");
        }
        logger.info(DASHES);

        logger.info(DASHES);
        CloudFormationHelper.destroyCloudFormationStack(ROLES_STACK);
        logger.info("This concludes the AWS IoT SiteWise Scenario");
        logger.info(DASHES);
    }

    private static void waitForInputToContinue(Scanner scanner) {
        while (true) {
            logger.info("");
            logger.info("Enter 'c' followed by <ENTER> to continue:");
            String input = scanner.nextLine();

            if (input.trim().equalsIgnoreCase("c")) {
                logger.info("Continuing with the program...");
```

```
                logger.info("");
                break;
            } else {
                logger.info("Invalid input. Please try again.");
            }
        }
    }

    public static void countdown(int totalSeconds) throws InterruptedException {
        for (int i = totalSeconds; i >= 0; i--) {
            int displayMinutes = i / 60;
            int displaySeconds = i % 60;
            System.out.printf("\r%02d:%02d", displayMinutes, displaySeconds);
            Thread.sleep(1000); // Wait for 1 second
        }
        System.out.println(); // Move to the next line after countdown
        logger.info("Countdown complete!");
    }
}
```

Una classe wrapper per i metodi SDK. AWS IoT SiteWise

```
public class SitewiseActions {

    private static final Logger logger =
 LoggerFactory.getLogger(SitewiseActions.class);

    private static IoTSiteWiseAsyncClient ioTSiteWiseAsyncClient;

    private static IoTSiteWiseAsyncClient getAsyncClient() {
        if (ioTSiteWiseAsyncClient == null) {
            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                .maxConcurrency(100)
                .connectionTimeout(Duration.ofSeconds(60))
                .readTimeout(Duration.ofSeconds(60))
                .writeTimeout(Duration.ofSeconds(60))
                .build();

            ClientOverrideConfiguration overrideConfig =
 ClientOverrideConfiguration.builder()
                .apiCallTimeout(Duration.ofMinutes(2))
                .apiCallAttemptTimeout(Duration.ofSeconds(90))
```

```
                .retryStrategy(RetryMode.STANDARD)
                .build();

            ioTSiteWiseAsyncClient = IoTSiteWiseAsyncClient.builder()
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return ioTSiteWiseAsyncClient;
    }


    /**
     * Creates an asset model.
     *
     * @param name the name of the asset model to create.
     * @return a {@link CompletableFuture} that represents a {@link
CreateAssetModelResponse} result. The calling code
     *         can attach callbacks, then handle the result or exception by calling
{@link CompletableFuture#join()} or
     *         {@link CompletableFuture#get()}.
     *         <p>
     *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps it
     *         available to the calling code as a {@link CompletionException}. By
calling
     *         {@link CompletionException#getCause()}, the calling code can access
the original exception.
     */
    public CompletableFuture<CreateAssetModelResponse> createAssetModelAsync(String
name) {
        PropertyType humidity = PropertyType.builder()
            .measurement(Measurement.builder().build())
            .build();

        PropertyType temperaturePropertyType = PropertyType.builder()
            .measurement(Measurement.builder().build())
            .build();

        AssetModelPropertyDefinition temperatureProperty =
AssetModelPropertyDefinition.builder()
            .name("Temperature")
            .dataType(PropertyDataType.DOUBLE)
            .type(temperaturePropertyType)
```

```
            .build();

        AssetModelPropertyDefinition humidityProperty =
AssetModelPropertyDefinition.builder()
            .name("Humidity")
            .dataType(PropertyDataType.DOUBLE)
            .type(humidity)
            .build();

        CreateAssetModelRequest createAssetModelRequest =
CreateAssetModelRequest.builder()
            .assetModelName(name)
            .assetModelDescription("This is my asset model")
            .assetModelProperties(temperatureProperty, humidityProperty)
            .build();

        return getAsyncClient().createAssetModel(createAssetModelRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    logger.error("Failed to create asset model: {} ",
exception.getCause().getMessage());
                }
            });
    }


    /**
     * Creates an asset with the specified name and asset model Id.
     *
     * @param assetName    the name of the asset to create.
     * @param assetModelId the Id of the asset model to associate with the asset.
     * @return a {@link CompletableFuture} that represents a {@link
CreateAssetResponse} result. The calling code can
     *         attach callbacks, then handle the result or exception by calling
{@link CompletableFuture#join()} or
     *         {@link CompletableFuture#get()}.
     *         <p>
     *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps it
     *         available to the calling code as a {@link CompletionException}. By
calling
     *         {@link CompletionException#getCause()}, the calling code can access
the original exception.
     */
```

```
   public CompletableFuture<CreateAssetResponse> createAssetAsync(String assetName,
String assetModelId) {
       CreateAssetRequest createAssetRequest = CreateAssetRequest.builder()
           .assetModelId(assetModelId)
           .assetDescription("Created using the AWS SDK for Java")
           .assetName(assetName)
           .build();

       return getAsyncClient().createAsset(createAssetRequest)
           .whenComplete((response, exception) -> {
               if (exception != null) {
                   logger.error("Failed to create asset: {}",
exception.getCause().getMessage());
               }
           });
   }

   /**
    * Sends data to the SiteWise service.
    *
    * @param assetId        the ID of the asset to which the data will be sent.
    * @param tempPropertyId the ID of the temperature property.
    * @param humidityPropId the ID of the humidity property.
    * @return a {@link CompletableFuture} that represents a {@link
BatchPutAssetPropertyValueResponse} result. The
    *         calling code can attach callbacks, then handle the result or
exception by calling
    *         {@link CompletableFuture#join()} or {@link CompletableFuture#get()}.
    *         <p>
    *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps it
    *         available to the calling code as a {@link CompletionException}. By
calling
    *         {@link CompletionException#getCause()}, the calling code can access
the original exception.
    */
   public CompletableFuture<BatchPutAssetPropertyValueResponse>
sendDataToSiteWiseAsync(String assetId, String tempPropertyId, String
humidityPropId) {
       Map<String, Double> sampleData = generateSampleData();
       long timestamp = Instant.now().toEpochMilli();

       TimeInNanos time = TimeInNanos.builder()
           .timeInSeconds(timestamp / 1000)
```

```
            .offsetInNanos((int) ((timestamp % 1000) * 1000000))
            .build();

        BatchPutAssetPropertyValueRequest request =
BatchPutAssetPropertyValueRequest.builder()
            .entries(Arrays.asList(
                PutAssetPropertyValueEntry.builder()
                    .entryId("entry-3")
                    .assetId(assetId)
                    .propertyId(tempPropertyId)
                    .propertyValues(Arrays.asList(
                        AssetPropertyValue.builder()
                            .value(Variant.builder()
                                .doubleValue(sampleData.get("Temperature"))
                                .build())
                            .timestamp(time)
                            .build()
                    ))
                    .build(),
                PutAssetPropertyValueEntry.builder()
                    .entryId("entry-4")
                    .assetId(assetId)
                    .propertyId(humidityPropId)
                    .propertyValues(Arrays.asList(
                        AssetPropertyValue.builder()
                            .value(Variant.builder()
                                .doubleValue(sampleData.get("Humidity"))
                                .build())
                            .timestamp(time)
                            .build()
                    ))
                    .build()
            ))
            .build();

        return getAsyncClient().batchPutAssetPropertyValue(request)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    logger.error("An exception occurred: {}",
exception.getCause().getMessage());
                }
            });
    }
```

```
   /**
    * Fetches the value of an asset property.
    *
    * @param propId  the ID of the asset property to fetch.
    * @param assetId the ID of the asset to fetch the property value for.
    * @return a {@link CompletableFuture} that represents a {@link Double} result.
The calling code can attach
    *         callbacks, then handle the result or exception by calling {@link
CompletableFuture#join()} or
    *         {@link CompletableFuture#get()}.
    *         <p>
    *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps
    *         it available to the calling code as a {@link CompletionException}. By
calling
    *         {@link CompletionException#getCause()}, the calling code can access
the original exception.
    */
   public CompletableFuture<Double> getAssetPropValueAsync(String propId, String
assetId) {
       GetAssetPropertyValueRequest assetPropertyValueRequest =
GetAssetPropertyValueRequest.builder()
               .propertyId(propId)
               .assetId(assetId)
               .build();

       return getAsyncClient().getAssetPropertyValue(assetPropertyValueRequest)
               .handle((response, exception) -> {
                   if (exception != null) {
                       logger.error("Error occurred while fetching property value:
{}.", exception.getCause().getMessage());
                       throw (CompletionException) exception;
                   }
                   return response.propertyValue().value().doubleValue();
               });
   }

   /**
    * Retrieves the property IDs associated with a specific asset model.
    *
    * @param assetModelId the ID of the asset model that defines the properties.
    * @return a {@link CompletableFuture} that represents a {@link Map} result that
associates the property name to the
```

```
    *         propert ID. The calling code can attach callbacks, then handle the
result or exception by calling
    *         {@link CompletableFuture#join()} or {@link CompletableFuture#get()}.
    *         <p>
    *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps
    *         it available to the calling code as a {@link CompletionException}. By
calling
    *         {@link CompletionException#getCause()}, the calling code can access
the original exception.
    */
   public CompletableFuture<Map<String, String>> getPropertyIds(String
assetModelId) {
       ListAssetModelPropertiesRequest modelPropertiesRequest =
ListAssetModelPropertiesRequest.builder().assetModelId(assetModelId).build();
       return getAsyncClient().listAssetModelProperties(modelPropertiesRequest)
           .handle((response, throwable) -> {
               if (response != null) {
                   return response.assetModelPropertySummaries().stream()
                       .collect(Collectors
                           .toMap(AssetModelPropertySummary::name,
AssetModelPropertySummary::id));
               } else {
                   logger.error("Error occurred while fetching property IDs: {}.",
throwable.getCause().getMessage());
                   throw (CompletionException) throwable;
               }
           });
   }

   /**
    * Deletes an asset.
    *
    * @param assetId the ID of the asset to be deleted.
    * @return a {@link CompletableFuture} that represents a {@link
DeleteAssetResponse} result. The calling code can
    *         attach callbacks, then handle the result or exception by calling
{@link CompletableFuture#join()} or
    *         {@link CompletableFuture#get()}.
    *         <p>
    *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps
    *         it available to the calling code as a {@link CompletionException}. By
calling
```

```
     *         {@link CompletionException#getCause()}, the calling code can access
the original exception.
     */
    public CompletableFuture<DeleteAssetResponse> deleteAssetAsync(String assetId) {
        DeleteAssetRequest deleteAssetRequest = DeleteAssetRequest.builder()
            .assetId(assetId)
            .build();

        return getAsyncClient().deleteAsset(deleteAssetRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    logger.error("An error occurred deleting asset with id: {}",
assetId);
                }
            });
    }

    /**
     * Deletes an Asset Model with the specified ID.
     *
     * @param assetModelId the ID of the Asset Model to delete.
     * @return a {@link CompletableFuture} that represents a {@link
DeleteAssetModelResponse} result. The calling code
     *         can attach callbacks, then handle the result or exception by calling
{@link CompletableFuture#join()} or
     *         {@link CompletableFuture#get()}.
     *         <p>
     *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps
     *         it available to the calling code as a {@link CompletionException}. By
calling
     *         {@link CompletionException#getCause()}, the calling code can access
the original exception.
     */
    public CompletableFuture<DeleteAssetModelResponse> deleteAssetModelAsync(String
assetModelId) {
        DeleteAssetModelRequest deleteAssetModelRequest =
DeleteAssetModelRequest.builder()
            .assetModelId(assetModelId)
            .build();

        return getAsyncClient().deleteAssetModel(deleteAssetModelRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
```

```
                    logger.error("Failed to delete asset model with ID:{}.",
exception.getMessage());
                }
            });
    }

    /**
     * Creates a new IoT SiteWise portal.
     *
     * @param portalName   the name of the portal to create.
     * @param iamRole      the IAM role ARN to use for the portal.
     * @param contactEmail the email address of the portal contact.
     * @return a {@link CompletableFuture} that represents a {@link String} result
of the portal ID. The calling code
     *         can attach callbacks, then handle the result or exception by calling
{@link CompletableFuture#join()} or
     *         {@link CompletableFuture#get()}.
     *         <p>
     *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps
     *         it available to the calling code as a {@link CompletionException}. By
calling
     *         {@link CompletionException#getCause()}, the calling code can access
the original exception.
     */
    public CompletableFuture<String> createPortalAsync(String portalName, String
iamRole, String contactEmail) {
        CreatePortalRequest createPortalRequest = CreatePortalRequest.builder()
            .portalName(portalName)
            .portalDescription("This is my custom IoT SiteWise portal.")
            .portalContactEmail(contactEmail)
            .roleArn(iamRole)
            .build();

        return getAsyncClient().createPortal(createPortalRequest)
            .handle((response, exception) -> {
                if (exception != null) {
                    logger.error("Failed to create portal: {} ",
exception.getCause().getMessage());
                    throw (CompletionException) exception;
                }
                return response.portalId();
            });
    }
```

```
   /**
    * Deletes a portal.
    *
    * @param portalId the ID of the portal to be deleted.
    * @return a {@link CompletableFuture} that represents a {@link
DeletePortalResponse}. The calling code can attach
    *         callbacks, then handle the result or exception by calling {@link
CompletableFuture#join()} or
    *         {@link CompletableFuture#get()}.
    *         <p>
    *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps
    *         it available to the calling code as a {@link CompletionException}. By
calling
    *         {@link CompletionException#getCause()}, the calling code can access
the original exception.
    */
   public CompletableFuture<DeletePortalResponse> deletePortalAsync(String
portalId) {
       DeletePortalRequest deletePortalRequest = DeletePortalRequest.builder()
           .portalId(portalId)
           .build();

       return getAsyncClient().deletePortal(deletePortalRequest)
           .whenComplete((response, exception) -> {
               if (exception != null) {
                   logger.error("Failed to delete portal with ID: {}. Error: {}",
portalId, exception.getCause().getMessage());
               }
           });
   }

   /**
    * Retrieves the asset model ID for the given asset model name.
    *
    * @param assetModelName the name of the asset model for the ID.
    * @return a {@link CompletableFuture} that represents a {@link String} result
of the asset model ID or null if the
    *         asset model cannot be found. The calling code can attach callbacks,
then handle the result or exception
    *         by calling {@link CompletableFuture#join()} or {@link
CompletableFuture#get()}.
    *         <p>
```

```
    *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps
    *         it available to the calling code as a {@link CompletionException}. By
calling
    *         {@link CompletionException#getCause()}, the calling code can access
the original exception.
    */
   public CompletableFuture<String> getAssetModelIdAsync(String assetModelName) {
       ListAssetModelsRequest listAssetModelsRequest =
ListAssetModelsRequest.builder().build();
       return getAsyncClient().listAssetModels(listAssetModelsRequest)
               .handle((listAssetModelsResponse, exception) -> {
                   if (exception != null) {
                       logger.error("Failed to retrieve Asset Model ID: {}",
exception.getCause().getMessage());
                       throw (CompletionException) exception;
                   }
                   for (AssetModelSummary assetModelSummary :
listAssetModelsResponse.assetModelSummaries()) {
                       if (assetModelSummary.name().equals(assetModelName)) {
                           return assetModelSummary.id();
                       }
                   }
                   return null;
               });
   }

   /**
    * Retrieves a portal's description.
    *
    * @param portalId the ID of the portal to describe.
    * @return a {@link CompletableFuture} that represents a {@link String} result
of the portal's start URL
    *         (see: {@link DescribePortalResponse#portalStartUrl()}). The calling
code can attach callbacks, then handle the
    *         result or exception by calling {@link CompletableFuture#join()} or
{@link CompletableFuture#get()}.
    *         <p>
    *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps
    *         it available to the calling code as a {@link CompletionException}. By
calling
    *         {@link CompletionException#getCause()}, the calling code can access
the original exception.
```

```
     */
    public CompletableFuture<String> describePortalAsync(String portalId) {
        DescribePortalRequest request = DescribePortalRequest.builder()
            .portalId(portalId)
            .build();

        return getAsyncClient().describePortal(request)
            .handle((response, exception) -> {
                if (exception != null) {
                   logger.error("An exception occurred retrieving the portal
description: {}", exception.getCause().getMessage());
                   throw (CompletionException) exception;
                }
                return response.portalStartUrl();
            });
    }


    /**
     * Creates a new IoT Sitewise gateway.
     *
     * @param gatewayName The name of the gateway to create.
     * @param myThing     The name of the core device thing to associate with the
gateway.
     * @return a {@link CompletableFuture} that represents a {@link String} result
of the gateways ID. The calling code
     *         can attach callbacks, then handle the result or exception by calling
{@link CompletableFuture#join()} or
     *         {@link CompletableFuture#get()}.
     *         <p>
     *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps
     *         it available to the calling code as a {@link CompletionException}. By
calling
     *         {@link CompletionException#getCause()}, the calling code can access
the original exception.
     */
    public CompletableFuture<String> createGatewayAsync(String gatewayName, String
myThing) {
        GreengrassV2 gg = GreengrassV2.builder()
            .coreDeviceThingName(myThing)
            .build();

        GatewayPlatform platform = GatewayPlatform.builder()
```

```
            .greengrassV2(gg)
            .build();

        Map<String, String> tag = new HashMap<>();
        tag.put("Environment", "Production");

        CreateGatewayRequest createGatewayRequest = CreateGatewayRequest.builder()
            .gatewayName(gatewayName)
            .gatewayPlatform(platform)
            .tags(tag)
            .build();

        return getAsyncClient().createGateway(createGatewayRequest)
            .handle((response, exception) -> {
                if (exception != null) {
                    logger.error("Error creating the gateway.");
                    throw (CompletionException) exception;
                }
                logger.info("The ARN of the gateway is {}" ,
response.gatewayArn());
                return response.gatewayId();
            });
    }

    /**
     * Deletes the specified gateway.
     *
     * @param gatewayId the ID of the gateway to delete.
     * @return a {@link CompletableFuture} that represents a {@link
DeleteGatewayResponse} result.. The calling code
     *         can attach callbacks, then handle the result or exception by calling
{@link CompletableFuture#join()} or
     *         {@link CompletableFuture#get()}.
     *         <p>
     *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps
     *         it available to the calling code as a {@link CompletionException}. By
calling
     *         {@link CompletionException#getCause()}, the calling code can access
the original exception.
     */
    public CompletableFuture<DeleteGatewayResponse> deleteGatewayAsync(String
gatewayId) {
        DeleteGatewayRequest deleteGatewayRequest = DeleteGatewayRequest.builder()
```

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

        return getAsyncClient().deleteGateway(deleteGatewayRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    logger.error("Failed to delete gateway: {}",
exception.getCause().getMessage());
                }
            });
    }

    /**
     * Describes the specified gateway.
     *
     * @param gatewayId the ID of the gateway to describe.
     * @return a {@link CompletableFuture} that represents a {@link
DescribeGatewayResponse} result. The calling code
     *         can attach callbacks, then handle the result or exception by calling
{@link CompletableFuture#join()} or
     *         {@link CompletableFuture#get()}.
     *         <p>
     *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps
     *         it available to the calling code as a {@link CompletionException}. By
calling
     *         {@link CompletionException#getCause()}, the calling code can access
the original exception.
     */
    public CompletableFuture<DescribeGatewayResponse> describeGatewayAsync(String
gatewayId) {
        DescribeGatewayRequest request = DescribeGatewayRequest.builder()
            .gatewayId(gatewayId)
            .build();

        return getAsyncClient().describeGateway(request)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    logger.error("An error occurred during the describeGateway
method: {}", exception.getCause().getMessage());
                }
            });
    }
```

```
    private static Map<String, Double> generateSampleData() {
        Map<String, Double> data = new HashMap<>();
        data.put("Temperature", 23.5);
        data.put("Humidity", 65.0);
        return data;
    }
}
```

Azioni

`BatchPutAssetPropertyValue`

Il seguente esempio di codice mostra come usare. `BatchPutAssetPropertyValue`

SDK per Java 2.x

> **Note**
>
> C'è 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 data to the SiteWise service.
     *
     * @param assetId        the ID of the asset to which the data will be sent.
     * @param tempPropertyId the ID of the temperature property.
     * @param humidityPropId the ID of the humidity property.
     * @return a {@link CompletableFuture} that represents a {@link
BatchPutAssetPropertyValueResponse} result. The
     *         calling code can attach callbacks, then handle the result or
exception by calling
     *         {@link CompletableFuture#join()} or {@link CompletableFuture#get()}.
     *         <p>
     *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps it
     *         available to the calling code as a {@link CompletionException}. By
calling
     *         {@link CompletionException#getCause()}, the calling code can access
the original exception.
```

```
     */
    public CompletableFuture<BatchPutAssetPropertyValueResponse>
sendDataToSiteWiseAsync(String assetId, String tempPropertyId, String
humidityPropId) {
        Map<String, Double> sampleData = generateSampleData();
        long timestamp = Instant.now().toEpochMilli();

        TimeInNanos time = TimeInNanos.builder()
            .timeInSeconds(timestamp / 1000)
            .offsetInNanos((int) ((timestamp % 1000) * 1000000))
            .build();

        BatchPutAssetPropertyValueRequest request =
BatchPutAssetPropertyValueRequest.builder()
            .entries(Arrays.asList(
                PutAssetPropertyValueEntry.builder()
                    .entryId("entry-3")
                    .assetId(assetId)
                    .propertyId(tempPropertyId)
                    .propertyValues(Arrays.asList(
                        AssetPropertyValue.builder()
                            .value(Variant.builder()
                                .doubleValue(sampleData.get("Temperature"))
                                .build())
                            .timestamp(time)
                            .build()
                    ))
                    .build(),
                PutAssetPropertyValueEntry.builder()
                    .entryId("entry-4")
                    .assetId(assetId)
                    .propertyId(humidityPropId)
                    .propertyValues(Arrays.asList(
                        AssetPropertyValue.builder()
                            .value(Variant.builder()
                                .doubleValue(sampleData.get("Humidity"))
                                .build())
                            .timestamp(time)
                            .build()
                    ))
                    .build()
            ))
            .build();
```

```
        return getAsyncClient().batchPutAssetPropertyValue(request)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    logger.error("An exception occurred: {}",
exception.getCause().getMessage());
                }
            });
    }
```

- Per i dettagli sull'API, BatchPutAssetPropertyValueconsulta AWS SDK for Java 2.x API Reference.

`CreateAsset`

Il seguente esempio di codice mostra come utilizzare`CreateAsset`.

SDK per Java 2.x

> **Note**
>
> C'è 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 asset with the specified name and asset model Id.
     *
     * @param assetName    the name of the asset to create.
     * @param assetModelId the Id of the asset model to associate with the asset.
     * @return a {@link CompletableFuture} that represents a {@link
CreateAssetResponse} result. The calling code can
     *         attach callbacks, then handle the result or exception by calling
{@link CompletableFuture#join()} or
     *         {@link CompletableFuture#get()}.
     *         <p>
     *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps it
     *         available to the calling code as a {@link CompletionException}. By
calling
```

```
     *         {@link CompletionException#getCause()}, the calling code can access
the original exception.
     */
    public CompletableFuture<CreateAssetResponse> createAssetAsync(String assetName,
String assetModelId) {
        CreateAssetRequest createAssetRequest = CreateAssetRequest.builder()
            .assetModelId(assetModelId)
            .assetDescription("Created using the AWS SDK for Java")
            .assetName(assetName)
            .build();

        return getAsyncClient().createAsset(createAssetRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    logger.error("Failed to create asset: {}",
exception.getCause().getMessage());
                }
            });
    }
```

- Per i dettagli sull'API, CreateAssetconsulta AWS SDK for Java 2.x API Reference.

`CreateAssetModel`

Il seguente esempio di codice mostra come utilizzare`CreateAssetModel`.

SDK per Java 2.x

> **Note**
>
> C'è 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 asset model.
     *
     * @param name the name of the asset model to create.
     * @return a {@link CompletableFuture} that represents a {@link
CreateAssetModelResponse} result. The calling code
```

```
    *         can attach callbacks, then handle the result or exception by calling
{@link CompletableFuture#join()} or
    *         {@link CompletableFuture#get()}.
    *         <p>
    *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps it
    *         available to the calling code as a {@link CompletionException}. By
calling
    *         {@link CompletionException#getCause()}, the calling code can access
the original exception.
    */
   public CompletableFuture<CreateAssetModelResponse> createAssetModelAsync(String
name) {
       PropertyType humidity = PropertyType.builder()
           .measurement(Measurement.builder().build())
           .build();

       PropertyType temperaturePropertyType = PropertyType.builder()
           .measurement(Measurement.builder().build())
           .build();

       AssetModelPropertyDefinition temperatureProperty =
AssetModelPropertyDefinition.builder()
           .name("Temperature")
           .dataType(PropertyDataType.DOUBLE)
           .type(temperaturePropertyType)
           .build();

       AssetModelPropertyDefinition humidityProperty =
AssetModelPropertyDefinition.builder()
           .name("Humidity")
           .dataType(PropertyDataType.DOUBLE)
           .type(humidity)
           .build();

       CreateAssetModelRequest createAssetModelRequest =
CreateAssetModelRequest.builder()
           .assetModelName(name)
           .assetModelDescription("This is my asset model")
           .assetModelProperties(temperatureProperty, humidityProperty)
           .build();

       return getAsyncClient().createAssetModel(createAssetModelRequest)
           .whenComplete((response, exception) -> {
```

```
                if (exception != null) {
                    logger.error("Failed to create asset model: {} ",
exception.getCause().getMessage());
                }
            });
    }
```

- Per i dettagli sull'API, CreateAssetModelconsulta AWS SDK for Java 2.x API Reference.

`CreateGateway`

Il seguente esempio di codice mostra come utilizzare`CreateGateway`.

SDK per Java 2.x

> **Note**
>
> C'è 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 a new IoT Sitewise gateway.
     *
     * @param gatewayName The name of the gateway to create.
     * @param myThing     The name of the core device thing to associate with the
gateway.
     * @return a {@link CompletableFuture} that represents a {@link String} result
of the gateways ID. The calling code
     *         can attach callbacks, then handle the result or exception by calling
{@link CompletableFuture#join()} or
     *         {@link CompletableFuture#get()}.
     *         <p>
     *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps
     *         it available to the calling code as a {@link CompletionException}. By
calling
     *         {@link CompletionException#getCause()}, the calling code can access
the original exception.
```

```
     */
    public CompletableFuture<String> createGatewayAsync(String gatewayName, String
myThing) {
        GreengrassV2 gg = GreengrassV2.builder()
            .coreDeviceThingName(myThing)
            .build();

        GatewayPlatform platform = GatewayPlatform.builder()
            .greengrassV2(gg)
            .build();

        Map<String, String> tag = new HashMap<>();
        tag.put("Environment", "Production");

        CreateGatewayRequest createGatewayRequest = CreateGatewayRequest.builder()
            .gatewayName(gatewayName)
            .gatewayPlatform(platform)
            .tags(tag)
            .build();

        return getAsyncClient().createGateway(createGatewayRequest)
            .handle((response, exception) -> {
                if (exception != null) {
                    logger.error("Error creating the gateway.");
                    throw (CompletionException) exception;
                }
                logger.info("The ARN of the gateway is {}" ,
response.gatewayArn());
                return response.gatewayId();
            });
    }
```

- Per i dettagli sull'API, CreateGatewayconsulta AWS SDK for Java 2.x API Reference.

`CreatePortal`

Il seguente esempio di codice mostra come utilizzare`CreatePortal`.

SDK per Java 2.x

> **Note**
>
> C'è 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 a new IoT SiteWise portal.
    *
    * @param portalName   the name of the portal to create.
    * @param iamRole      the IAM role ARN to use for the portal.
    * @param contactEmail the email address of the portal contact.
    * @return a {@link CompletableFuture} that represents a {@link String} result
of the portal ID. The calling code
    *         can attach callbacks, then handle the result or exception by calling
{@link CompletableFuture#join()} or
    *         {@link CompletableFuture#get()}.
    *         <p>
    *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps
    *         it available to the calling code as a {@link CompletionException}. By
calling
    *         {@link CompletionException#getCause()}, the calling code can access
the original exception.
    */
   public CompletableFuture<String> createPortalAsync(String portalName, String
iamRole, String contactEmail) {
       CreatePortalRequest createPortalRequest = CreatePortalRequest.builder()
           .portalName(portalName)
           .portalDescription("This is my custom IoT SiteWise portal.")
           .portalContactEmail(contactEmail)
           .roleArn(iamRole)
           .build();

       return getAsyncClient().createPortal(createPortalRequest)
           .handle((response, exception) -> {
               if (exception != null) {
                   logger.error("Failed to create portal: {} ",
exception.getCause().getMessage());
                   throw (CompletionException) exception;
```

```
                }
                return response.portalId();
            });
    }
```

- Per i dettagli sull'API, CreatePortalconsulta AWS SDK for Java 2.x API Reference.

`DeleteAsset`

Il seguente esempio di codice mostra come utilizzare`DeleteAsset`.

SDK per Java 2.x

> **Note**
>
> C'è 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 asset.
     *
     * @param assetId the ID of the asset to be deleted.
     * @return a {@link CompletableFuture} that represents a {@link
DeleteAssetResponse} result. The calling code can
     *         attach callbacks, then handle the result or exception by calling
{@link CompletableFuture#join()} or
     *         {@link CompletableFuture#get()}.
     *         <p>
     *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps
     *         it available to the calling code as a {@link CompletionException}. By
calling
     *         {@link CompletionException#getCause()}, the calling code can access
the original exception.
     */
    public CompletableFuture<DeleteAssetResponse> deleteAssetAsync(String assetId) {
        DeleteAssetRequest deleteAssetRequest = DeleteAssetRequest.builder()
            .assetId(assetId)
            .build();
```

```
        return getAsyncClient().deleteAsset(deleteAssetRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    logger.error("An error occurred deleting asset with id: {}",
assetId);
                }
            });
    }
```

- Per i dettagli sull'API, DeleteAssetconsulta AWS SDK for Java 2.x API Reference.

`DeleteAssetModel`

Il seguente esempio di codice mostra come utilizzare`DeleteAssetModel`.

SDK per Java 2.x

> **Note**
>
> C'è 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 Asset Model with the specified ID.
     *
     * @param assetModelId the ID of the Asset Model to delete.
     * @return a {@link CompletableFuture} that represents a {@link
DeleteAssetModelResponse} result. The calling code
     *         can attach callbacks, then handle the result or exception by calling
{@link CompletableFuture#join()} or
     *         {@link CompletableFuture#get()}.
     *         <p>
     *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps
     *         it available to the calling code as a {@link CompletionException}. By
calling
     *         {@link CompletionException#getCause()}, the calling code can access
the original exception.
```

```
    */
   public CompletableFuture<DeleteAssetModelResponse> deleteAssetModelAsync(String
assetModelId) {
       DeleteAssetModelRequest deleteAssetModelRequest =
DeleteAssetModelRequest.builder()
           .assetModelId(assetModelId)
           .build();

       return getAsyncClient().deleteAssetModel(deleteAssetModelRequest)
           .whenComplete((response, exception) -> {
               if (exception != null) {
                   logger.error("Failed to delete asset model with ID:{}.",
exception.getMessage());
               }
           });
   }
```

- Per i dettagli sull'API, DeleteAssetModelconsulta AWS SDK for Java 2.x API Reference.

`DeleteGateway`

Il seguente esempio di codice mostra come utilizzare`DeleteGateway`.

SDK per Java 2.x

> **Note**
>
> C'è 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 specified gateway.
    *
    * @param gatewayId the ID of the gateway to delete.
    * @return a {@link CompletableFuture} that represents a {@link
DeleteGatewayResponse} result.. The calling code
    *         can attach callbacks, then handle the result or exception by calling
{@link CompletableFuture#join()} or
    *         {@link CompletableFuture#get()}.
    *         <p>
```

```
    *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps
    *         it available to the calling code as a {@link CompletionException}. By
calling
    *         {@link CompletionException#getCause()}, the calling code can access
the original exception.
    */
   public CompletableFuture<DeleteGatewayResponse> deleteGatewayAsync(String
gatewayId) {
       DeleteGatewayRequest deleteGatewayRequest = DeleteGatewayRequest.builder()
           .gatewayId(gatewayId)
           .build();

       return getAsyncClient().deleteGateway(deleteGatewayRequest)
           .whenComplete((response, exception) -> {
               if (exception != null) {
                   logger.error("Failed to delete gateway: {}",
exception.getCause().getMessage());
               }
           });
   }
```

- Per i dettagli sull'API, DeleteGatewayconsulta AWS SDK for Java 2.x API Reference.

`DeletePortal`

Il seguente esempio di codice mostra come utilizzare`DeletePortal`.

SDK per Java 2.x

> **Note**
>
> C'è 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 portal.
    *
    * @param portalId the ID of the portal to be deleted.
```

```
    * @return a {@link CompletableFuture} that represents a {@link
DeletePortalResponse}. The calling code can attach
    *         callbacks, then handle the result or exception by calling {@link
CompletableFuture#join()} or
    *         {@link CompletableFuture#get()}.
    *         <p>
    *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps
    *         it available to the calling code as a {@link CompletionException}. By
calling
    *         {@link CompletionException#getCause()}, the calling code can access
the original exception.
    */
   public CompletableFuture<DeletePortalResponse> deletePortalAsync(String
portalId) {
       DeletePortalRequest deletePortalRequest = DeletePortalRequest.builder()
           .portalId(portalId)
           .build();

       return getAsyncClient().deletePortal(deletePortalRequest)
           .whenComplete((response, exception) -> {
               if (exception != null) {
                   logger.error("Failed to delete portal with ID: {}. Error: {}",
portalId, exception.getCause().getMessage());
               }
           });
   }
```

- Per i dettagli sull'API, DeletePortalconsulta AWS SDK for Java 2.x API Reference.

`DescribeAssetModel`

Il seguente esempio di codice mostra come utilizzare`DescribeAssetModel`.

SDK per Java 2.x

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
    /**
     * Retrieves the property IDs associated with a specific asset model.
     *
     * @param assetModelId the ID of the asset model that defines the properties.
     * @return a {@link CompletableFuture} that represents a {@link Map} result that
associates the property name to the
     *         propert ID. The calling code can attach callbacks, then handle the
result or exception by calling
     *         {@link CompletableFuture#join()} or {@link CompletableFuture#get()}.
     *         <p>
     *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps
     *         it available to the calling code as a {@link CompletionException}. By
calling
     *         {@link CompletionException#getCause()}, the calling code can access
the original exception.
     */
    public CompletableFuture<Map<String, String>> getPropertyIds(String
assetModelId) {
        ListAssetModelPropertiesRequest modelPropertiesRequest =
ListAssetModelPropertiesRequest.builder().assetModelId(assetModelId).build();
        return getAsyncClient().listAssetModelProperties(modelPropertiesRequest)
            .handle((response, throwable) -> {
                if (response != null) {
                    return response.assetModelPropertySummaries().stream()
                        .collect(Collectors
                            .toMap(AssetModelPropertySummary::name,
AssetModelPropertySummary::id));
                } else {
                    logger.error("Error occurred while fetching property IDs: {}.",
throwable.getCause().getMessage());
                    throw (CompletionException) throwable;
                }
            });
    }
```

- Per i dettagli sull'API, DescribeAssetModelconsulta AWS SDK for Java 2.x API Reference.

`DescribeGateway`

Il seguente esempio di codice mostra come utilizzare`DescribeGateway`.

SDK per Java 2.x

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
   /**
    * Describes the specified gateway.
    *
    * @param gatewayId the ID of the gateway to describe.
    * @return a {@link CompletableFuture} that represents a {@link
DescribeGatewayResponse} result. The calling code
    *         can attach callbacks, then handle the result or exception by calling
{@link CompletableFuture#join()} or
    *         {@link CompletableFuture#get()}.
    *         <p>
    *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps
    *         it available to the calling code as a {@link CompletionException}. By
calling
    *         {@link CompletionException#getCause()}, the calling code can access
the original exception.
    */
   public CompletableFuture<DescribeGatewayResponse> describeGatewayAsync(String
gatewayId) {
       DescribeGatewayRequest request = DescribeGatewayRequest.builder()
           .gatewayId(gatewayId)
           .build();

       return getAsyncClient().describeGateway(request)
           .whenComplete((response, exception) -> {
               if (exception != null) {
                   logger.error("An error occurred during the describeGateway
method: {}", exception.getCause().getMessage());
               }
           });
   }
```

- Per i dettagli sull'API, DescribeGatewayconsulta AWS SDK for Java 2.x API Reference.

`DescribePortal`

Il seguente esempio di codice mostra come utilizzare`DescribePortal`.

SDK per Java 2.x

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
   /**
    * Retrieves a portal's description.
    *
    * @param portalId the ID of the portal to describe.
    * @return a {@link CompletableFuture} that represents a {@link String} result
of the portal's start URL
    *         (see: {@link DescribePortalResponse#portalStartUrl()}). The calling
code can attach callbacks, then handle the
    *         result or exception by calling {@link CompletableFuture#join()} or
{@link CompletableFuture#get()}.
    *         <p>
    *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps
    *         it available to the calling code as a {@link CompletionException}. By
calling
    *         {@link CompletionException#getCause()}, the calling code can access
the original exception.
    */
   public CompletableFuture<String> describePortalAsync(String portalId) {
       DescribePortalRequest request = DescribePortalRequest.builder()
           .portalId(portalId)
           .build();

       return getAsyncClient().describePortal(request)
           .handle((response, exception) -> {
               if (exception != null) {
                   logger.error("An exception occurred retrieving the portal
description: {}", exception.getCause().getMessage());
                   throw (CompletionException) exception;
               }
               return response.portalStartUrl();
```

```
            });
    }
```

- Per i dettagli sull'API, DescribePortalconsulta AWS SDK for Java 2.x API Reference.

`GetAssetPropertyValue`

Il seguente esempio di codice mostra come utilizzare`GetAssetPropertyValue`.

SDK per Java 2.x

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
    /**
     * Fetches the value of an asset property.
     *
     * @param propId  the ID of the asset property to fetch.
     * @param assetId the ID of the asset to fetch the property value for.
     * @return a {@link CompletableFuture} that represents a {@link Double} result.
The calling code can attach
     *         callbacks, then handle the result or exception by calling {@link
CompletableFuture#join()} or
     *         {@link CompletableFuture#get()}.
     *         <p>
     *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps
     *         it available to the calling code as a {@link CompletionException}. By
calling
     *         {@link CompletionException#getCause()}, the calling code can access
the original exception.
     */
    public CompletableFuture<Double> getAssetPropValueAsync(String propId, String
assetId) {
        GetAssetPropertyValueRequest assetPropertyValueRequest =
GetAssetPropertyValueRequest.builder()
                .propertyId(propId)
```

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

        return getAsyncClient().getAssetPropertyValue(assetPropertyValueRequest)
                .handle((response, exception) -> {
                    if (exception != null) {
                        logger.error("Error occurred while fetching property value:
{}.", exception.getCause().getMessage());
                        throw (CompletionException) exception;
                    }
                    return response.propertyValue().value().doubleValue();
                });
    }
```

- Per i dettagli sull'API, GetAssetPropertyValueconsulta AWS SDK for Java 2.x API Reference.

`ListAssetModels`

Il seguente esempio di codice mostra come utilizzare`ListAssetModels`.

SDK per Java 2.x

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
    /**
     * Retrieves the asset model ID for the given asset model name.
     *
     * @param assetModelName the name of the asset model for the ID.
     * @return a {@link CompletableFuture} that represents a {@link String} result
of the asset model ID or null if the
     *         asset model cannot be found. The calling code can attach callbacks,
then handle the result or exception
     *         by calling {@link CompletableFuture#join()} or {@link
CompletableFuture#get()}.
     *         <p>
     *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps
```

```
     *         it available to the calling code as a {@link CompletionException}. By
calling
     *         {@link CompletionException#getCause()}, the calling code can access
the original exception.
     */
    public CompletableFuture<String> getAssetModelIdAsync(String assetModelName) {
        ListAssetModelsRequest listAssetModelsRequest =
ListAssetModelsRequest.builder().build();
        return getAsyncClient().listAssetModels(listAssetModelsRequest)
                .handle((listAssetModelsResponse, exception) -> {
                    if (exception != null) {
                        logger.error("Failed to retrieve Asset Model ID: {}",
exception.getCause().getMessage());
                        throw (CompletionException) exception;
                    }
                    for (AssetModelSummary assetModelSummary :
listAssetModelsResponse.assetModelSummaries()) {
                        if (assetModelSummary.name().equals(assetModelName)) {
                            return assetModelSummary.id();
                        }
                    }
                    return null;
                });
    }
```

- Per i dettagli sull'API, ListAssetModelsconsulta 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 nozioni di base sono esempi di codice che mostrano come eseguire le operazioni essenziali all'interno di un servizio.

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.

Ogni esempio include un collegamento al codice sorgente completo, 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, ListKeyspacesconsulta AWS SDK for Java 2.x API Reference.

Argomenti

- Nozioni di base
- Azioni

Nozioni di base

Informazioni di base

L'esempio di codice seguente mostra come:

- Crea uno spazio di chiavi 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 keyspaces using a paginator.");
        listKeyspacesPaginator(keyClient);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. Create a table with a simple movie data schema and
enable point-in-time recovery.");
        createTable(keyClient, keyspaceName, tableName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. Check for the table to be in an Active state.");
        Thread.sleep(6000);
        checkTable(keyClient, keyspaceName, tableName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6. List all tables in the keyspace.");
        listTables(keyClient, keyspaceName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Use a Cassandra driver to insert some records into
the Movie table.");
        Thread.sleep(6000);
        loadData(session, fileName, keyspaceName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("8. Get all records from the Movie table.");
        getMovieData(session, keyspaceName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("9. Get a specific Movie.");
        getSpecificMovie(session, keyspaceName);
        System.out.println(DASHES);
```

```
        System.out.println(DASHES);
        System.out.println("10. Get a UTC timestamp for the current time.");
        ZonedDateTime utc = ZonedDateTime.now(ZoneOffset.UTC);
        System.out.println("DATETIME = " + Date.from(utc.toInstant()));
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("11. Update the table schema to add a watched Boolean
column.");
        updateTable(keyClient, keyspaceName, tableName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("12. Update an item as watched.");
        Thread.sleep(10000); // Wait 10 secs for the update.
        updateRecord(session, keyspaceName, titleUpdate, yearUpdate);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("13. Query for items with watched = True.");
        getWatchedData(session, keyspaceName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("14. Restore the table back to the previous state using
the timestamp.");
        System.out.println("Note that the restore operation can take up to 20
minutes.");
        restoreTable(keyClient, keyspaceName, utc);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("15. Check for completion of the restore action.");
        Thread.sleep(5000);
        checkRestoredTable(keyClient, keyspaceName, "MovieRestore");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("16. Delete both tables.");
        deleteTable(keyClient, keyspaceName, tableName);
        deleteTable(keyClient, keyspaceName, tableNameRestore);
        System.out.println(DASHES);

        System.out.println(DASHES);
```

```
        System.out.println("17. Confirm that both tables are deleted.");
        checkTableDelete(keyClient, keyspaceName, tableName);
        checkTableDelete(keyClient, keyspaceName, tableNameRestore);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("18. Delete the keyspace.");
        deleteKeyspace(keyClient, keyspaceName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("The scenario has completed successfully.");
        System.out.println(DASHES);
    }

    public static void deleteKeyspace(KeyspacesClient keyClient, String
keyspaceName) {
        try {
            DeleteKeyspaceRequest deleteKeyspaceRequest =
DeleteKeyspaceRequest.builder()
                    .keyspaceName(keyspaceName)
                    .build();

            keyClient.deleteKeyspace(deleteKeyspaceRequest);

        } catch (KeyspacesException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void checkTableDelete(KeyspacesClient keyClient, String
keyspaceName, String tableName)
            throws InterruptedException {
        try {
            String status;
            GetTableResponse response;
            GetTableRequest tableRequest = GetTableRequest.builder()
                    .keyspaceName(keyspaceName)
                    .tableName(tableName)
                    .build();

            // Keep looping until table cannot be found and a
ResourceNotFoundException is
```

```
            // thrown.
            while (true) {
                response = keyClient.getTable(tableRequest);
                status = response.statusAsString();
                System.out.println(". The table status is " + status);
                Thread.sleep(500);
            }

        } catch (ResourceNotFoundException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }
        System.out.println("The table is deleted");
    }

    public static void deleteTable(KeyspacesClient keyClient, String keyspaceName,
String tableName) {
        try {
            DeleteTableRequest tableRequest = DeleteTableRequest.builder()
                    .keyspaceName(keyspaceName)
                    .tableName(tableName)
                    .build();

            keyClient.deleteTable(tableRequest);

        } catch (KeyspacesException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void checkRestoredTable(KeyspacesClient keyClient, String
keyspaceName, String tableName)
            throws InterruptedException {
        try {
            boolean tableStatus = false;
            String status;
            GetTableResponse response = null;
            GetTableRequest tableRequest = GetTableRequest.builder()
                    .keyspaceName(keyspaceName)
                    .tableName(tableName)
                    .build();

            while (!tableStatus) {
                response = keyClient.getTable(tableRequest);
```

```
                status = response.statusAsString();
                System.out.println("The table status is " + status);

                if (status.compareTo("ACTIVE") == 0) {
                    tableStatus = true;
                }
                Thread.sleep(500);
            }

            List<ColumnDefinition> cols = response.schemaDefinition().allColumns();
            for (ColumnDefinition def : cols) {
                System.out.println("The column name is " + def.name());
                System.out.println("The column type is " + def.type());
            }

        } catch (KeyspacesException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void restoreTable(KeyspacesClient keyClient, String keyspaceName,
ZonedDateTime utc) {
        try {
            Instant myTime = utc.toInstant();
            RestoreTableRequest restoreTableRequest = RestoreTableRequest.builder()
                    .restoreTimestamp(myTime)
                    .sourceTableName("Movie")
                    .targetKeyspaceName(keyspaceName)
                    .targetTableName("MovieRestore")
                    .sourceKeyspaceName(keyspaceName)
                    .build();

            RestoreTableResponse response =
keyClient.restoreTable(restoreTableRequest);
            System.out.println("The ARN of the restored table is " +
response.restoredTableARN());

        } catch (KeyspacesException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

```
   public static void getWatchedData(CqlSession session, String keyspaceName) {
       ResultSet resultSet = session
               .execute("SELECT * FROM \"" + keyspaceName + "\".\"Movie\" WHERE
watched = true ALLOW FILTERING;");
       resultSet.forEach(item -> {
           System.out.println("The Movie title is " + item.getString("title"));
           System.out.println("The Movie year is " + item.getInt("year"));
           System.out.println("The plot is " + item.getString("plot"));
       });
   }

   public static void updateRecord(CqlSession session, String keySpace, String
titleUpdate, int yearUpdate) {
       String sqlStatement = "UPDATE \"" + keySpace
               + "\".\"Movie\" SET watched=true WHERE title = :k0 AND year = :k1;";
       BatchStatementBuilder builder =
BatchStatement.builder(DefaultBatchType.UNLOGGED);
       builder.setConsistencyLevel(ConsistencyLevel.LOCAL_QUORUM);
       PreparedStatement preparedStatement = session.prepare(sqlStatement);
       builder.addStatement(preparedStatement.boundStatementBuilder()
               .setString("k0", titleUpdate)
               .setInt("k1", yearUpdate)
               .build());

       BatchStatement batchStatement = builder.build();
       session.execute(batchStatement);
   }

   public static void updateTable(KeyspacesClient keyClient, String keySpace,
String tableName) {
       try {
           ColumnDefinition def = ColumnDefinition.builder()
                   .name("watched")
                   .type("boolean")
                   .build();

           UpdateTableRequest tableRequest = UpdateTableRequest.builder()
                   .keyspaceName(keySpace)
                   .tableName(tableName)
                   .addColumns(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> 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);
        }
    }

    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

Azioni

`CreateKeyspace`

Il seguente esempio di codice mostra come usare`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, CreateKeyspaceconsulta 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, CreateTableconsulta 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, DeleteKeyspaceconsulta 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, DeleteTableconsulta 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, GetKeyspaceconsulta 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, GetTableconsulta 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, ListKeyspacesconsulta 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, ListTablesconsulta 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, RestoreTableconsulta 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)
                   .addColumns(def)
                   .build();

           keyClient.updateTable(tableRequest);

       } catch (KeyspacesException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }
```

- Per i dettagli sull'API, UpdateTableconsulta AWS SDK for Java 2.x API Reference.

Esempi di Kinesis 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 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.

Ogni esempio include un collegamento al codice sorgente completo, 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, CreateStreamconsulta 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, DeleteStreamconsulta 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, GetRecordsconsulta 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, PutRecordconsulta 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 nozioni di base sono esempi di codice che mostrano come eseguire le operazioni essenziali all'interno di un servizio.

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.

Ogni esempio include un collegamento al codice sorgente completo, in cui è possibile trovare istruzioni su come configurare ed eseguire il codice nel contesto.

Nozioni di base

Salve AWS Key Management Service

L'esempio di codice seguente mostra come iniziare a utilizzare AWS Key Management Service.

SDK per Java 2.x

> **Note**
>
> C'è 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.kms.KmsAsyncClient;
import software.amazon.awssdk.services.kms.model.ListKeysRequest;
import software.amazon.awssdk.services.kms.paginators.ListKeysPublisher;
import java.util.concurrent.CompletableFuture;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * 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) {
        listAllKeys();
    }

    public static void listAllKeys() {
        KmsAsyncClient kmsAsyncClient = KmsAsyncClient.builder()
            .build();
        ListKeysRequest listKeysRequest = ListKeysRequest.builder()
            .limit(15)
            .build();
```

```
        /*
         * The `subscribe` method is required when using paginator methods in the
AWS SDK
         * because paginator methods return an instance of a `ListKeysPublisher`,
which is
         * based on a reactive stream. This allows asynchronous retrieval of
paginated
         * results as they become available. By subscribing to the stream, we can
process
         * each page of results as they are emitted.
         */
        ListKeysPublisher keysPublisher =
kmsAsyncClient.listKeysPaginator(listKeysRequest);
        CompletableFuture<Void> future = keysPublisher
            .subscribe(r -> r.keys().forEach(key ->
                System.out.println("The key ARN is: " + key.keyArn() + ". The key Id
is: " + key.keyId())))
            .whenComplete((result, exception) -> {
                if (exception != null) {
                    System.err.println("Error occurred: " + exception.getMessage());
                } else {
                    System.out.println("Successfully listed all keys.");
                }
            });

        try {
            future.join();
        } catch (Exception e) {
            System.err.println("Failed to list keys: " + e.getMessage());
        }
    }
}
```

- Per i dettagli sull'API, ListKeysconsulta AWS SDK for Java 2.x API Reference.

Argomenti

- Nozioni di base
- Azioni

Nozioni di base

Informazioni di base

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.

Esegui uno scenario al prompt dei comandi.

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.kms.model.AlreadyExistsException;
import software.amazon.awssdk.services.kms.model.DisabledException;
import software.amazon.awssdk.services.kms.model.EnableKeyRotationResponse;
import software.amazon.awssdk.services.kms.model.KmsException;
import software.amazon.awssdk.services.kms.model.NotFoundException;
import software.amazon.awssdk.services.kms.model.RevokeGrantResponse;
import java.util.List;
import java.util.Scanner;
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.CompletionException;
```

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * 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 String accountId = "";

    private static final Logger logger = LoggerFactory.getLogger(KMSScenario.class);

    static KMSActions kmsActions = new KMSActions();

    static Scanner scanner = new Scanner(System.in);

    static String aliasName = "alias/dev-encryption-key";

    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) {
            logger.info(usage);
            return;
        }
        String granteePrincipal = args[0];
        String policyName = "default";

        accountId = kmsActions.getAccountId();
        String keyDesc = "Created by the AWS KMS API";

        logger.info(DASHES);
        logger.info("""
            Welcome to the AWS Key Management SDK Basics 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 Basics scenario creates two key types:

            - A symmetric encryption key is used to encrypt and decrypt data.
            - An asymmetric key used to digitally sign data.

            Let's get started...
            """);
        waitForInputToContinue(scanner);

        try {
        // Run the methods that belong to this scenario.
        String targetKeyId = runScenario(granteePrincipal, keyDesc, policyName);
        requestDeleteResources(aliasName, targetKeyId);

        } catch (Throwable rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof KmsException kmsEx) {
                logger.info("KMS error occurred: Error message: {}, Error code {}",
kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
        }
    }

    private static String runScenario(String granteePrincipal, String keyDesc,
String policyName) throws Throwable {
        logger.info(DASHES);
        logger.info("1. Create a symmetric KMS key\n");
        logger.info("First, the program will creates a symmetric KMS key that you
can used to encrypt and decrypt data.");
        waitForInputToContinue(scanner);
        String targetKeyId;
        try {
```

```
            CompletableFuture<String> futureKeyId =
kmsActions.createKeyAsync(keyDesc);
            targetKeyId = futureKeyId.join();
            logger.info("A symmetric key was successfully created " + targetKeyId);

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof KmsException kmsEx) {
                logger.info("KMS error occurred: Error message: {}, Error code {}",
kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);

        logger.info(DASHES);
        logger.info("""
            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.
             """);
        waitForInputToContinue(scanner);
        boolean isEnabled;
        try {
            CompletableFuture<Boolean> futureIsKeyEnabled =
kmsActions.isKeyEnabledAsync(targetKeyId);
            isEnabled = futureIsKeyEnabled.join();
            logger.info("Is the key enabled? {}", isEnabled);

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof KmsException kmsEx) {
                logger.info("KMS error occurred: Error message: {}, Error code {}",
kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            throw cause;
        }
```

```
        if (!isEnabled)
            try {
                CompletableFuture<Void> future =
kmsActions.enableKeyAsync(targetKeyId);
                future.join();

            } catch (RuntimeException rt) {
                Throwable cause = rt.getCause();
                if (cause instanceof KmsException kmsEx) {
                    logger.info("KMS error occurred: Error message: {}, Error code
{}", kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
                } else {
                    logger.info("An unexpected error occurred: " + rt.getMessage());
                }
                throw cause;
            }
        waitForInputToContinue(scanner);

        logger.info(DASHES);
        logger.info("3. Encrypt data using the symmetric KMS key");
        String plaintext = "Hello, AWS KMS!";
        logger.info("""
            One of the main uses of symmetric keys is to encrypt and decrypt data.
            Next, the code encrypts the string {} with the SYMMETRIC_DEFAULT
encryption algorithm.
            """, plaintext);
        waitForInputToContinue(scanner);
        SdkBytes encryptedData;
        try {
            CompletableFuture<SdkBytes> future =
kmsActions.encryptDataAsync(targetKeyId, plaintext);
            encryptedData = future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof DisabledException kmsDisabledEx) {
                logger.info("KMS error occurred due to a disabled
key: Error message: {}, Error code {}", kmsDisabledEx.getMessage(),
kmsDisabledEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            deleteKey(targetKeyId);
            throw cause;
```

```
        }
        waitForInputToContinue(scanner);

        logger.info(DASHES);
        logger.info("4. Create an alias");
        logger.info("""

            The alias name should be prefixed with 'alias/'.
            The default, 'alias/dev-encryption-key'.
             """);
        waitForInputToContinue(scanner);

        try {
            CompletableFuture<Void> future =
kmsActions.createCustomAliasAsync(targetKeyId, aliasName);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof AlreadyExistsException kmsExistsEx) {
                if (kmsExistsEx.getMessage().contains("already exists")) {
                    logger.info("The alias '" + aliasName + "' already exists.
Moving on...");
                }
            } else {
                logger.error("An unexpected error occurred: " + rt.getMessage(),
rt);
                deleteKey(targetKeyId);
                throw cause;
            }
        }
        waitForInputToContinue(scanner);

        logger.info(DASHES);
        logger.info("5. List all of your aliases");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Object> future = kmsActions.listAllAliasesAsync();
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof KmsException kmsEx) {
```

```
                logger.info("KMS error occurred: Error message: {}, Error code {}",
kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            deleteAliasName(aliasName);
            deleteKey(targetKeyId);
            throw cause;
        }
        waitForInputToContinue(scanner);

        logger.info(DASHES);
        logger.info("6. Enable automatic rotation of the KMS key");
        logger.info("""

            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);
        try {
            CompletableFuture<EnableKeyRotationResponse> future =
kmsActions.enableKeyRotationAsync(targetKeyId);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof KmsException kmsEx) {
                logger.info("KMS error occurred: Error message: {}, Error code {}",
kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            deleteAliasName(aliasName);
            deleteKey(targetKeyId);
            throw cause;
        }
        waitForInputToContinue(scanner);

        logger.info(DASHES);
        logger.info("""
            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 = null;
        try {
            CompletableFuture<String> futureGrantId =
kmsActions.grantKeyAsync(targetKeyId, granteePrincipal);
            grantId = futureGrantId.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof KmsException kmsEx) {
                logger.info("KMS error occurred: Error message: {}, Error code {}",
kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            deleteKey(targetKeyId);
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("8. List grants for the KMS key");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Object> future =
kmsActions.displayGrantIdsAsync(targetKeyId);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof KmsException kmsEx) {
                logger.info("KMS error occurred: Error message: {}, Error code {}",
kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
```

```
            }
            deleteAliasName(aliasName);
            deleteKey(targetKeyId);
            throw cause;
        }
        waitForInputToContinue(scanner);

        logger.info(DASHES);
        logger.info("9. Revoke the grant");
        logger.info("""
            The revocation of a grant immediately removes the permissions and access
that the grant had provided.
            This means that any principal (user, role, or service) that was granted
access to perform specific
            KMS operations on a KMS key will no longer be able to perform those
operations.
            """);
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<RevokeGrantResponse> future =
kmsActions.revokeKeyGrantAsync(targetKeyId, grantId);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof KmsException kmsEx) {
                if (kmsEx.getMessage().contains("Grant does not exist")) {
                    logger.info("The grant ID '" + grantId + "' does not exist.
Moving on...");
                } else {
                    logger.info("KMS error occurred: Error message: {}, Error code
{}", kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
                    throw cause;
                }
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
                deleteAliasName(aliasName);
                deleteKey(targetKeyId);
                throw cause;
            }
        }
        waitForInputToContinue(scanner);

        logger.info(DASHES);
```

```
        logger.info("10. Decrypt the data\n");
        logger.info("""
            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 decryptedData = "";
        try {
            CompletableFuture<String> future =
kmsActions.decryptDataAsync(encryptedData, targetKeyId);
            decryptedData = future.join();
            logger.info("Decrypted data: " + decryptedData);

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof KmsException kmsEx) {
                logger.info("KMS error occurred: Error message: {}, Error code {}",
kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            deleteAliasName(aliasName);
            deleteKey(targetKeyId);
            throw cause;
        }
        logger.info("Decrypted text is: " + decryptedData);
        waitForInputToContinue(scanner);

        logger.info(DASHES);
        logger.info("11. Replace a key policy\n");
        logger.info("""
            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::0000000000:root"},
                "Action": "kms:*",
                "Resource": "*"
                }]
            """);

        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Boolean> future =
kmsActions.replacePolicyAsync(targetKeyId, policyName, accountId);
            boolean success = future.join();
            if (success) {
                logger.info("Key policy replacement succeeded.");
            } else {
                logger.error("Key policy replacement failed.");
            }

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof KmsException kmsEx) {
                logger.info("KMS error occurred: Error message: {}, Error code {}",
kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            deleteAliasName(aliasName);
            deleteKey(targetKeyId);
            throw cause;
        }
        waitForInputToContinue(scanner);

        logger.info(DASHES);
        logger.info("12. Get the key policy\n");
        logger.info("The next bit of code that runs gets the key policy to make sure
it exists.");
        waitForInputToContinue(scanner);
```

```
        try {
            CompletableFuture<String> future =
kmsActions.getKeyPolicyAsync(targetKeyId, policyName);
            String policy = future.join();
            if (!policy.isEmpty()) {
                logger.info("Retrieved policy: " + policy);
            }

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof KmsException kmsEx) {
                logger.info("KMS error occurred: Error message: {}, Error code {}",
kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            deleteAliasName(aliasName);
            deleteKey(targetKeyId);
            throw cause;
        }
        waitForInputToContinue(scanner);

        logger.info(DASHES);
        logger.info("13. Create an asymmetric KMS key and sign your data\n");
        logger.info("""
             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);
        try {
            CompletableFuture<Boolean> future = kmsActions.signVerifyDataAsync();
            boolean success = future.join();
            if (success) {
                logger.info("Sign and verify data operation succeeded.");
            } else {
                logger.error("Sign and verify data operation failed.");
            }
```

```
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof KmsException kmsEx) {
                logger.info("KMS error occurred: Error message: {}, Error code {}",
kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            deleteAliasName(aliasName);
            deleteKey(targetKeyId);
            throw cause;
        }
        waitForInputToContinue(scanner);

        logger.info(DASHES);
        logger.info("14. Tag your symmetric KMS Key\n");
        logger.info("""
            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);
        try {
            CompletableFuture<Void> future = kmsActions.tagKMSKeyAsync(targetKeyId);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof KmsException kmsEx) {
                logger.info("KMS error occurred: Error message: {}, Error code {}",
kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            deleteAliasName(aliasName);
            deleteKey(targetKeyId);
            throw cause;
        }
        waitForInputToContinue(scanner);
        return targetKeyId;
    }
```

```
   // Deletes KMS resources with user input.
   private static void requestDeleteResources(String aliasName, String targetKeyId)
{
       logger.info(DASHES);
       logger.info("15. Schedule the deletion of the KMS key\n");
       logger.info("""
           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.
           """);
       logger.info("Would you like to delete the Key Management resources? (y/n)");
       String delAns = scanner.nextLine().trim();
       if (delAns.equalsIgnoreCase("y")) {
           logger.info("You selected to delete the AWS KMS resources.");
           waitForInputToContinue(scanner);
           try {
               CompletableFuture<Void> future =
kmsActions.deleteSpecificAliasAsync(aliasName);
               future.join();

           } catch (RuntimeException rt) {
               Throwable cause = rt.getCause();
               if (cause instanceof KmsException kmsEx) {
                   logger.info("KMS error occurred: Error message: {}, Error code
{}", kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
               } else {
                   logger.info("An unexpected error occurred: " + rt.getMessage());
               }
           }
           waitForInputToContinue(scanner);
           try {
               CompletableFuture<Void> future =
kmsActions.disableKeyAsync(targetKeyId);
               future.join();

           } catch (RuntimeException rt) {
```

```
                Throwable cause = rt.getCause();
                if (cause instanceof KmsException kmsEx) {
                    logger.info("KMS error occurred: Error message: {}, Error code
{}", kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
                } else {
                    logger.info("An unexpected error occurred: " + rt.getMessage());
                }
            }

            try {
                CompletableFuture<Void> future =
kmsActions.deleteKeyAsync(targetKeyId);
                future.join();

            } catch (RuntimeException rt) {
                Throwable cause = rt.getCause();
                if (cause instanceof KmsException kmsEx) {
                    logger.info("KMS error occurred: Error message: {}, Error code
{}", kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
                } else {
                    logger.info("An unexpected error occurred: " + rt.getMessage());
                }
            }

        } else {
            logger.info("The Key Management resources will not be deleted");
        }

        logger.info(DASHES);
        logger.info("This concludes the AWS Key Management SDK scenario");
        logger.info(DASHES);
    }

    // This method is invoked from Exceptions to clean up the resources.
    private static void deleteKey(String targetKeyId) {
        try {
            CompletableFuture<Void> future =
kmsActions.disableKeyAsync(targetKeyId);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof KmsException kmsEx) {
```

```
                logger.info("KMS error occurred: Error message: {}, Error code {}",
kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
        }

        try {
            CompletableFuture<Void> future = kmsActions.deleteKeyAsync(targetKeyId);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof KmsException kmsEx) {
                logger.info("KMS error occurred: Error message: {}, Error code {}",
kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
        }
    }

    // This method is invoked from Exceptions to clean up the resources.
    private static void deleteAliasName(String aliasName) {
        try {
            CompletableFuture<Void> future =
kmsActions.deleteSpecificAliasAsync(aliasName);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof KmsException kmsEx) {
                logger.info("KMS error occurred: Error message: {}, Error code {}",
kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
        }
    }

    private static void waitForInputToContinue(Scanner scanner) {
        while (true) {
            logger.info("");
            logger.info("Enter 'c' followed by <ENTER> to continue:");
```

```
            String input = scanner.nextLine();

            if (input.trim().equalsIgnoreCase("c")) {
                logger.info("Continuing with the program...");
                logger.info("");
                break;
            } else {
                // Handle invalid input.
                logger.info("Invalid input. Please try again.");
            }
        }
    }
}
```

Definisci una classe che racchiuda le azioni KMS.

```
public class KMSActions {
    private static final Logger logger = LoggerFactory.getLogger(KMSActions.class);
    private static KmsAsyncClient kmsAsyncClient;

    /**
     * Retrieves an asynchronous AWS Key Management Service (KMS) client.
     * <p>
     * This method creates and returns a singleton instance of the KMS async client,
 with the following configurations:
     * <ul>
     *   <li>Max concurrency: 100</li>
     *   <li>Connection timeout: 60 seconds</li>
     *   <li>Read timeout: 60 seconds</li>
     *   <li>Write timeout: 60 seconds</li>
     *   <li>API call timeout: 2 minutes</li>
     *   <li>API call attempt timeout: 90 seconds</li>
     *   <li>Retry policy: up to 3 retries</li>
     *   <li>Credentials provider: environment variable credentials provider</li>
     * </ul>
     * <p>
     * If the client instance has already been created, it is returned instead of
 creating a new one.
     *
     * @return the KMS async client instance
     */
    private static KmsAsyncClient getAsyncClient() {
```

```
        if (kmsAsyncClient == null) {
            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                .maxConcurrency(100)
                .connectionTimeout(Duration.ofSeconds(60))
                .readTimeout(Duration.ofSeconds(60))
                .writeTimeout(Duration.ofSeconds(60))
                .build();

            ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
                .apiCallTimeout(Duration.ofMinutes(2))
                .apiCallAttemptTimeout(Duration.ofSeconds(90))
                .retryPolicy(RetryPolicy.builder()
                    .numRetries(3)
                    .build())
                .build();

            kmsAsyncClient = KmsAsyncClient.builder()
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return kmsAsyncClient;
    }

    /**
     * Creates a new symmetric encryption key asynchronously.
     *
     * @param keyDesc the description of the key to be created
     * @return a {@link CompletableFuture} that completes with the ID of the newly
created key
     * @throws RuntimeException if an error occurs while creating the key
     */
    public CompletableFuture<String> createKeyAsync(String keyDesc) {
        CreateKeyRequest keyRequest = CreateKeyRequest.builder()
            .description(keyDesc)
            .keySpec(KeySpec.SYMMETRIC_DEFAULT)
            .keyUsage(KeyUsageType.ENCRYPT_DECRYPT)
            .build();

        return getAsyncClient().createKey(keyRequest)
            .thenApply(resp -> resp.keyMetadata().keyId())
            .exceptionally(ex -> {
```

```
                throw new RuntimeException("An error occurred while creating the
key: " + ex.getMessage(), ex);
            });
    }

    /**
     * Asynchronously checks if a specified key is enabled.
     *
     * @param keyId the ID of the key to check
     * @return a {@link CompletableFuture} that, when completed, indicates whether
the key is enabled or not
     *
     * @throws RuntimeException if an exception occurs while checking the key state
     */
    public CompletableFuture<Boolean> isKeyEnabledAsync(String keyId) {
        DescribeKeyRequest keyRequest = DescribeKeyRequest.builder()
            .keyId(keyId)
            .build();

        CompletableFuture<DescribeKeyResponse> responseFuture =
getAsyncClient().describeKey(keyRequest);
        return responseFuture.whenComplete((resp, ex) -> {
            if (resp != null) {
                KeyState keyState = resp.keyMetadata().keyState();
                if (keyState == KeyState.ENABLED) {
                    logger.info("The key is enabled.");
                } else {
                    logger.info("The key is not enabled. Key state: {}", keyState);
                }
            } else {
                throw new RuntimeException(ex);
            }
        }).thenApply(resp -> resp.keyMetadata().keyState() == KeyState.ENABLED);
    }

    /**
     * Asynchronously enables the specified key.
     *
     * @param keyId the ID of the key to enable
     * @return a {@link CompletableFuture} that completes when the key has been
enabled
     */
    public CompletableFuture<Void> enableKeyAsync(String keyId) {
        EnableKeyRequest enableKeyRequest = EnableKeyRequest.builder()
```

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

        CompletableFuture<EnableKeyResponse> responseFuture =
getAsyncClient().enableKey(enableKeyRequest);
        responseFuture.whenComplete((response, exception) -> {
            if (exception == null) {
                logger.info("Key with ID [{}] has been enabled.", keyId);
            } else {
                if (exception instanceof KmsException kmsEx) {
                    throw new RuntimeException("KMS error occurred while enabling
key: " + kmsEx.getMessage(), kmsEx);
                } else {
                    throw new RuntimeException("An unexpected error occurred while
enabling key: " + exception.getMessage(), exception);
                }
            }
        });

        return responseFuture.thenApply(response -> null);
    }

    /**
     * Encrypts the given text asynchronously using the specified KMS client and key
ID.
     *
     * @param keyId the ID of the KMS key to use for encryption
     * @param text the text to encrypt
     * @return a CompletableFuture that completes with the encrypted data as an
SdkBytes object
     */
    public CompletableFuture<SdkBytes> encryptDataAsync(String keyId, String text) {
        SdkBytes myBytes = SdkBytes.fromUtf8String(text);
        EncryptRequest encryptRequest = EncryptRequest.builder()
            .keyId(keyId)
            .plaintext(myBytes)
            .build();

        CompletableFuture<EncryptResponse> responseFuture =
getAsyncClient().encrypt(encryptRequest).toCompletableFuture();
        return responseFuture.whenComplete((response, ex) -> {
            if (response != null) {
                String algorithm = response.encryptionAlgorithm().toString();
```

```
                logger.info("The string was encrypted with algorithm {}.",
algorithm);
            } else {
                throw new RuntimeException(ex);
            }
        }).thenApply(EncryptResponse::ciphertextBlob);
    }

    /**
     * Creates a custom alias for the specified target key asynchronously.
     *
     * @param targetKeyId the ID of the target key for the alias
     * @param aliasName   the name of the alias to create
     * @return a {@link CompletableFuture} that completes when the alias creation
operation is finished
     */
    public CompletableFuture<Void> createCustomAliasAsync(String targetKeyId, String
aliasName) {
        CreateAliasRequest aliasRequest = CreateAliasRequest.builder()
            .aliasName(aliasName)
            .targetKeyId(targetKeyId)
            .build();

        CompletableFuture<CreateAliasResponse> responseFuture =
getAsyncClient().createAlias(aliasRequest);
        responseFuture.whenComplete((response, exception) -> {
            if (exception == null) {
                logger.info("{} was successfully created.", aliasName);
            } else {
                if (exception instanceof ResourceExistsException) {
                    logger.info("Alias [{}] already exists. Moving on...",
aliasName);
                } else if (exception instanceof KmsException kmsEx) {
                    throw new RuntimeException("KMS error occurred while creating
alias: " + kmsEx.getMessage(), kmsEx);
                } else {
                    throw new RuntimeException("An unexpected error occurred while
creating alias: " + exception.getMessage(), exception);
                }
            }
        });

        return responseFuture.thenApply(response -> null);
    }
```

```
   /**
    * Asynchronously lists all the aliases in the current AWS account.
    *
    * @return a {@link CompletableFuture} that completes when the list of aliases
has been processed
    */
   public CompletableFuture<Object> listAllAliasesAsync() {
       ListAliasesRequest aliasesRequest = ListAliasesRequest.builder()
           .limit(15)
           .build();

       ListAliasesPublisher paginator =
getAsyncClient().listAliasesPaginator(aliasesRequest);
       return paginator.subscribe(response -> {
               response.aliases().forEach(alias ->
                   logger.info("The alias name is: " + alias.aliasName())
               );
           })
           .thenApply(v -> null)
           .exceptionally(ex -> {
               if (ex.getCause() instanceof KmsException) {
                   KmsException e = (KmsException) ex.getCause();
                   throw new RuntimeException("A KMS exception occurred: " +
e.getMessage());
               } else {
                   throw new RuntimeException("An unexpected error occurred: " +
ex.getMessage());
               }
           });
   }

   /**
    * Enables key rotation asynchronously for the specified key ID.
    *
    * @param keyId the ID of the key for which to enable key rotation
    * @return a CompletableFuture that represents the asynchronous operation of
enabling key rotation
    * @throws RuntimeException if there was an error enabling key rotation, either
due to a KMS exception or an unexpected error
    */
   public CompletableFuture<EnableKeyRotationResponse>
enableKeyRotationAsync(String keyId) {
```

```
        EnableKeyRotationRequest enableKeyRotationRequest =
EnableKeyRotationRequest.builder()
            .keyId(keyId)
            .build();

        CompletableFuture<EnableKeyRotationResponse> responseFuture =
getAsyncClient().enableKeyRotation(enableKeyRotationRequest);
        responseFuture.whenComplete((response, exception) -> {
            if (exception == null) {
                logger.info("Key rotation has been enabled for key with id [{}]",
keyId);
            } else {
                if (exception instanceof KmsException kmsEx) {
                    throw new RuntimeException("Failed to enable key rotation: " +
kmsEx.getMessage(), kmsEx);
                } else {
                    throw new RuntimeException("An unexpected error occurred: " +
exception.getMessage(), exception);
                }
            }
        });

        return responseFuture;
    }

    /**
     * Grants permissions to a specified principal on a customer master key (CMK)
asynchronously.
     *
     * @param keyId             The unique identifier for the customer master key
(CMK) that the grant applies to.
     * @param granteePrincipal  The principal that is given permission to perform
the operations that the grant permits on the CMK.
     * @return A {@link CompletableFuture} that, when completed, contains the ID of
the created grant.
     * @throws RuntimeException If an error occurs during the grant creation
process.
     */
    public CompletableFuture<String> grantKeyAsync(String keyId, String
granteePrincipal) {
        List<GrantOperation> grantPermissions = List.of(
            GrantOperation.ENCRYPT,
            GrantOperation.DECRYPT,
            GrantOperation.DESCRIBE_KEY
```

```
        );

        CreateGrantRequest grantRequest = CreateGrantRequest.builder()
            .keyId(keyId)
            .name("grant1")
            .granteePrincipal(granteePrincipal)
            .operations(grantPermissions)
            .build();

        CompletableFuture<CreateGrantResponse> responseFuture =
getAsyncClient().createGrant(grantRequest);
        responseFuture.whenComplete((response, ex) -> {
            if (ex == null) {
                logger.info("Grant created successfully with ID: " +
response.grantId());
            } else {
                if (ex instanceof KmsException kmsEx) {
                    throw new RuntimeException("Failed to create grant: " +
kmsEx.getMessage(), kmsEx);
                } else {
                    throw new RuntimeException("An unexpected error occurred: " +
ex.getMessage(), ex);
                }
            }
        });

        return responseFuture.thenApply(CreateGrantResponse::grantId);
    }

    /**
     * Asynchronously displays the grant IDs for the specified key ID.
     *
     * @param keyId the ID of the AWS KMS key for which to list the grants
     * @return a {@link CompletableFuture} that, when completed, will be null if
the operation succeeded, or will throw a {@link RuntimeException} if the operation
failed
     * @throws RuntimeException if there was an error listing the grants, either due
to an {@link KmsException} or an unexpected error
     */
    public CompletableFuture<Object> displayGrantIdsAsync(String keyId) {
        ListGrantsRequest grantsRequest = ListGrantsRequest.builder()
            .keyId(keyId)
            .limit(15)
            .build();
```

```
       ListGrantsPublisher paginator =
getAsyncClient().listGrantsPaginator(grantsRequest);
       return paginator.subscribe(response -> {
               response.grants().forEach(grant -> {
                   logger.info("The grant Id is: " + grant.grantId());
               });
           })
           .thenApply(v -> null)
           .exceptionally(ex -> {
               Throwable cause = ex.getCause();
               if (cause instanceof KmsException) {
                   throw new RuntimeException("Failed to list grants: " +
cause.getMessage(), cause);
               } else {
                   throw new RuntimeException("An unexpected error occurred: " +
cause.getMessage(), cause);
               }
           });
   }

   /**
    * Revokes a grant for the specified AWS KMS key asynchronously.
    *
    * @param keyId   The ID or key ARN of the AWS KMS key.
    * @param grantId The identifier of the grant to be revoked.
    * @return A {@link CompletableFuture} representing the asynchronous operation
of revoking the grant.
    *         The {@link CompletableFuture} will complete with a {@link
RevokeGrantResponse} object
    *         if the operation is successful, or with a {@code null} value if an
error occurs.
    */
   public CompletableFuture<RevokeGrantResponse> revokeKeyGrantAsync(String keyId,
String grantId) {
       RevokeGrantRequest grantRequest = RevokeGrantRequest.builder()
           .keyId(keyId)
           .grantId(grantId)
           .build();

       CompletableFuture<RevokeGrantResponse> responseFuture =
getAsyncClient().revokeGrant(grantRequest);
       responseFuture.whenComplete((response, exception) -> {
           if (exception == null) {
```

```
                logger.info("Grant ID: [" + grantId + "] was successfully
revoked!");
            } else {
                if (exception instanceof KmsException kmsEx) {
                    if (kmsEx.getMessage().contains("Grant does not exist")) {
                        logger.info("The grant ID '" + grantId + "' does not exist.
Moving on...");
                    } else {
                        throw new RuntimeException("KMS error occurred: " +
kmsEx.getMessage(), kmsEx);
                    }
                } else {
                    throw new RuntimeException("An unexpected error occurred: " +
exception.getMessage(), exception);
                }
            }
        });

        return responseFuture;
    }


    /**
     * Asynchronously decrypts the given encrypted data using the specified key ID.
     *
     * @param encryptedData The encrypted data to be decrypted.
     * @param keyId The ID of the key to be used for decryption.
     * @return A CompletableFuture that, when completed, will contain the decrypted
data as a String.
     *         If an error occurs during the decryption process, the
CompletableFuture will complete
     *         exceptionally with the error, and the method will return an empty
String.
     */
    public CompletableFuture<String> decryptDataAsync(SdkBytes encryptedData, String
keyId) {
        DecryptRequest decryptRequest = DecryptRequest.builder()
            .ciphertextBlob(encryptedData)
            .keyId(keyId)
            .build();

        CompletableFuture<DecryptResponse> responseFuture =
getAsyncClient().decrypt(decryptRequest);
        responseFuture.whenComplete((decryptResponse, exception) -> {
```

```
            if (exception == null) {
                logger.info("Data decrypted successfully for key ID: " + keyId);
            } else {
                if (exception instanceof KmsException kmsEx) {
                    throw new RuntimeException("KMS error occurred while decrypting
data: " + kmsEx.getMessage(), kmsEx);
                } else {
                    throw new RuntimeException("An unexpected error occurred while
decrypting data: " + exception.getMessage(), exception);
                }
            }
        });

        return responseFuture.thenApply(decryptResponse ->
decryptResponse.plaintext().asString(StandardCharsets.UTF_8));
    }

    /**
     * Asynchronously replaces the policy for the specified KMS key.
     *
     * @param keyId       the ID of the KMS key to replace the policy for
     * @param policyName  the name of the policy to be replaced
     * @param accountId   the AWS account ID to be used in the policy
     * @return a {@link CompletableFuture} that completes with a boolean indicating
     *         whether the policy replacement was successful or not
     */
    public CompletableFuture<Boolean> replacePolicyAsync(String keyId, String
policyName, String accountId) {
        String policy = """
    {
      "Version": "2012-10-17",
      "Statement": [{
        "Effect": "Allow",
        "Principal": {"AWS": "arn:aws:iam::%s:root"},
        "Action": "kms:*",
        "Resource": "*"
      }]
    }
    """.formatted(accountId);

        PutKeyPolicyRequest keyPolicyRequest = PutKeyPolicyRequest.builder()
            .keyId(keyId)
            .policyName(policyName)
            .policy(policy)
```

```
            .build();

        // First, get the current policy to check if it exists
        return getAsyncClient().getKeyPolicy(r ->
r.keyId(keyId).policyName(policyName))
            .thenCompose(response -> {
                logger.info("Current policy exists. Replacing it...");
                return getAsyncClient().putKeyPolicy(keyPolicyRequest);
            })
            .thenApply(putPolicyResponse -> {
                logger.info("The key policy has been replaced.");
                return true;
            })
            .exceptionally(throwable -> {
                if (throwable.getCause() instanceof LimitExceededException) {
                    logger.error("Cannot replace policy, as only one policy is
allowed per key.");
                    return false;
                }
                throw new RuntimeException("Error replacing policy", throwable);
            });
    }


    /**
     * Asynchronously retrieves the key policy for the specified key ID and policy
name.
     *
     * @param keyId       the ID of the AWS KMS key for which to retrieve the policy
     * @param policyName the name of the key policy to retrieve
     * @return a {@link CompletableFuture} that, when completed, contains the key
policy as a {@link String}
     */
    public CompletableFuture<String> getKeyPolicyAsync(String keyId, String
policyName) {
        GetKeyPolicyRequest policyRequest = GetKeyPolicyRequest.builder()
            .keyId(keyId)
            .policyName(policyName)
            .build();

        return getAsyncClient().getKeyPolicy(policyRequest)
            .thenApply(response -> {
                String policy = response.policy();
                logger.info("The response is: " + policy);
```

```
                return policy;
            })
            .exceptionally(ex -> {
                throw new RuntimeException("Failed to get key policy", ex);
            });
    }

    /**
     * Asynchronously signs and verifies data using AWS KMS.
     *
     * <p>The method performs the following steps:
     * <ol>
     *     <li>Creates an AWS KMS key with the specified key spec, key usage, and
 origin.</li>
     *     <li>Signs the provided message using the created KMS key and the RSASSA-
PSS-SHA-256 algorithm.</li>
     *     <li>Verifies the signature of the message using the created KMS key and
 the RSASSA-PSS-SHA-256 algorithm.</li>
     * </ol>
     *
     * @return a {@link CompletableFuture} that completes with the result of the
 signature verification,
     *         {@code true} if the signature is valid, {@code false} otherwise.
     * @throws KmsException if any error occurs during the KMS operations.
     * @throws RuntimeException if an unexpected error occurs.
     */
    public CompletableFuture<Boolean> signVerifyDataAsync() {
        String signMessage = "Here is the message that will be digitally signed";

        // Create an AWS KMS key used to digitally sign data.
        CreateKeyRequest createKeyRequest = CreateKeyRequest.builder()
            .keySpec(KeySpec.RSA_2048)
            .keyUsage(KeyUsageType.SIGN_VERIFY)
            .origin(OriginType.AWS_KMS)
            .build();

        return getAsyncClient().createKey(createKeyRequest)
            .thenCompose(createKeyResponse -> {
                String keyId = createKeyResponse.keyMetadata().keyId();

                SdkBytes messageBytes = SdkBytes.fromString(signMessage,
 Charset.defaultCharset());
                SignRequest signRequest = SignRequest.builder()
                    .keyId(keyId)
```

```
                    .message(messageBytes)
                    .signingAlgorithm(SigningAlgorithmSpec.RSASSA_PSS_SHA_256)
                    .build();

                return getAsyncClient().sign(signRequest)
                    .thenCompose(signResponse -> {
                        byte[] signedBytes = signResponse.signature().asByteArray();

                        VerifyRequest verifyRequest = VerifyRequest.builder()
                            .keyId(keyId)

.message(SdkBytes.fromByteArray(signMessage.getBytes(Charset.defaultCharset())))

.signature(SdkBytes.fromByteBuffer(ByteBuffer.wrap(signedBytes)))

.signingAlgorithm(SigningAlgorithmSpec.RSASSA_PSS_SHA_256)
                            .build();

                        return getAsyncClient().verify(verifyRequest)
                            .thenApply(verifyResponse -> {
                                return (boolean) verifyResponse.signatureValid();
                            });
                    });
            })
            .exceptionally(throwable -> {
               throw new RuntimeException("Failed to sign or verify data",
throwable);
            });
    }

    /**
     * Asynchronously tags a KMS key with a specific tag.
     *
     * @param keyId the ID of the KMS key to be tagged
     * @return a {@link CompletableFuture} that completes when the tagging operation
is finished
     */
    public CompletableFuture<Void> tagKMSKeyAsync(String keyId) {
        Tag tag = Tag.builder()
            .tagKey("Environment")
            .tagValue("Production")
            .build();

        TagResourceRequest tagResourceRequest = TagResourceRequest.builder()
```

```
            .keyId(keyId)
            .tags(tag)
            .build();

        return getAsyncClient().tagResource(tagResourceRequest)
            .thenRun(() -> {
                logger.info("{} key was tagged", keyId);
            })
            .exceptionally(throwable -> {
                throw new RuntimeException("Failed to tag the KMS key", throwable);
            });
    }

    /**
     * Deletes a specific KMS alias asynchronously.
     *
     * @param aliasName the name of the alias to be deleted
     * @return a {@link CompletableFuture} representing the asynchronous operation
of deleting the specified alias
     */
    public CompletableFuture<Void> deleteSpecificAliasAsync(String aliasName) {
        DeleteAliasRequest deleteAliasRequest = DeleteAliasRequest.builder()
            .aliasName(aliasName)
            .build();

        return getAsyncClient().deleteAlias(deleteAliasRequest)
            .thenRun(() -> {
                logger.info("Alias {} has been deleted successfully", aliasName);
            })
            .exceptionally(throwable -> {
                throw new RuntimeException("Failed to delete alias: " + aliasName,
throwable);
            });
    }

    /**
     * Asynchronously disables the specified AWS Key Management Service (KMS) key.
     *
     * @param keyId the ID or Amazon Resource Name (ARN) of the KMS key to be
disabled
     * @return a CompletableFuture that, when completed, indicates that the key has
been disabled successfully
     */
    public CompletableFuture<Void> disableKeyAsync(String keyId) {
```

```
        DisableKeyRequest keyRequest = DisableKeyRequest.builder()
            .keyId(keyId)
            .build();

        return getAsyncClient().disableKey(keyRequest)
            .thenRun(() -> {
                logger.info("Key {} has been disabled successfully",keyId);
            })
            .exceptionally(throwable -> {
                throw new RuntimeException("Failed to disable key: " + keyId,
throwable);
            });
    }

    /**
     * Deletes a KMS key asynchronously.
     *
     * <p><strong>Warning:</strong> 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
becomes unrecoverable.
     * This means that any files, databases, or other data that were encrypted using
the deleted KMS key
     * will become permanently inaccessible. Exercise extreme caution when deleting
KMS keys.</p>
     *
     * @param keyId the ID of the KMS key to delete
     * @return a {@link CompletableFuture} that completes when the key deletion is
scheduled
     */
    public CompletableFuture<Void> deleteKeyAsync(String keyId) {
        ScheduleKeyDeletionRequest deletionRequest =
ScheduleKeyDeletionRequest.builder()
            .keyId(keyId)
            .pendingWindowInDays(7)
            .build();

        return getAsyncClient().scheduleKeyDeletion(deletionRequest)
            .thenRun(() -> {
                logger.info("Key {} will be deleted in 7 days", keyId);
            })
            .exceptionally(throwable -> {
                throw new RuntimeException("Failed to schedule key deletion for key
ID: " + keyId, throwable);
```

```
            });
    }


    public 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 .
  - CreateAlias
  - CreateGrant
  - CreateKey
  - Decrypt
  - DescribeKey
  - DisableKey
  - EnableKey
  - Encrypt
  - GetKeyPolicy
  - ListAliases
  - ListGrants
  - ListKeys
  - RevokeGrant
  - ScheduleKeyDeletion
  - Sign
  - TagResource

Azioni

`CreateAlias`

Il seguente esempio di codice mostra come usare. `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.

```
   /**
    * Creates a custom alias for the specified target key asynchronously.
    *
    * @param targetKeyId the ID of the target key for the alias
    * @param aliasName   the name of the alias to create
    * @return a {@link CompletableFuture} that completes when the alias creation
operation is finished
    */
   public CompletableFuture<Void> createCustomAliasAsync(String targetKeyId, String
aliasName) {
       CreateAliasRequest aliasRequest = CreateAliasRequest.builder()
           .aliasName(aliasName)
           .targetKeyId(targetKeyId)
           .build();

       CompletableFuture<CreateAliasResponse> responseFuture =
getAsyncClient().createAlias(aliasRequest);
       responseFuture.whenComplete((response, exception) -> {
           if (exception == null) {
               logger.info("{} was successfully created.", aliasName);
           } else {
               if (exception instanceof ResourceExistsException) {
                   logger.info("Alias [{}] already exists. Moving on...",
aliasName);
               } else if (exception instanceof KmsException kmsEx) {
                   throw new RuntimeException("KMS error occurred while creating
alias: " + kmsEx.getMessage(), kmsEx);
               } else {
```

```
                    throw new RuntimeException("An unexpected error occurred while
creating alias: " + exception.getMessage(), exception);
                }
            }
        });

        return responseFuture.thenApply(response -> null);
    }
```

- Per i dettagli sull'API, CreateAliasconsulta 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.

```
    /**
     * Grants permissions to a specified principal on a customer master key (CMK)
asynchronously.
     *
     * @param keyId             The unique identifier for the customer master key
(CMK) that the grant applies to.
     * @param granteePrincipal  The principal that is given permission to perform
the operations that the grant permits on the CMK.
     * @return A {@link CompletableFuture} that, when completed, contains the ID of
the created grant.
     * @throws RuntimeException If an error occurs during the grant creation
process.
     */
    public CompletableFuture<String> grantKeyAsync(String keyId, String
granteePrincipal) {
        List<GrantOperation> grantPermissions = List.of(
            GrantOperation.ENCRYPT,
```

```
            GrantOperation.DECRYPT,
            GrantOperation.DESCRIBE_KEY
        );

        CreateGrantRequest grantRequest = CreateGrantRequest.builder()
            .keyId(keyId)
            .name("grant1")
            .granteePrincipal(granteePrincipal)
            .operations(grantPermissions)
            .build();

        CompletableFuture<CreateGrantResponse> responseFuture =
getAsyncClient().createGrant(grantRequest);
        responseFuture.whenComplete((response, ex) -> {
            if (ex == null) {
                logger.info("Grant created successfully with ID: " +
response.grantId());
            } else {
                if (ex instanceof KmsException kmsEx) {
                    throw new RuntimeException("Failed to create grant: " +
kmsEx.getMessage(), kmsEx);
                } else {
                    throw new RuntimeException("An unexpected error occurred: " +
ex.getMessage(), ex);
                }
            }
        });

        return responseFuture.thenApply(CreateGrantResponse::grantId);
    }
```

- Per i dettagli sull'API, CreateGrantconsulta 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.

```
    /**
     * Creates a new symmetric encryption key asynchronously.
     *
     * @param keyDesc the description of the key to be created
     * @return a {@link CompletableFuture} that completes with the ID of the newly
created key
     * @throws RuntimeException if an error occurs while creating the key
     */
    public CompletableFuture<String> createKeyAsync(String keyDesc) {
        CreateKeyRequest keyRequest = CreateKeyRequest.builder()
            .description(keyDesc)
            .keySpec(KeySpec.SYMMETRIC_DEFAULT)
            .keyUsage(KeyUsageType.ENCRYPT_DECRYPT)
            .build();

        return getAsyncClient().createKey(keyRequest)
            .thenApply(resp -> resp.keyMetadata().keyId())
            .exceptionally(ex -> {
                throw new RuntimeException("An error occurred while creating the
key: " + ex.getMessage(), ex);
            });
    }
```

- Per i dettagli sull'API, CreateKeyconsulta 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.

```
    /**
     * Asynchronously decrypts the given encrypted data using the specified key ID.
     *
     * @param encryptedData The encrypted data to be decrypted.
     * @param keyId The ID of the key to be used for decryption.
     * @return A CompletableFuture that, when completed, will contain the decrypted
data as a String.
     *         If an error occurs during the decryption process, the
CompletableFuture will complete
     *         exceptionally with the error, and the method will return an empty
String.
     */
    public CompletableFuture<String> decryptDataAsync(SdkBytes encryptedData, String
keyId) {
        DecryptRequest decryptRequest = DecryptRequest.builder()
            .ciphertextBlob(encryptedData)
            .keyId(keyId)
            .build();

        CompletableFuture<DecryptResponse> responseFuture =
getAsyncClient().decrypt(decryptRequest);
        responseFuture.whenComplete((decryptResponse, exception) -> {
            if (exception == null) {
                logger.info("Data decrypted successfully for key ID: " + keyId);
            } else {
                if (exception instanceof KmsException kmsEx) {
                    throw new RuntimeException("KMS error occurred while decrypting
data: " + kmsEx.getMessage(), kmsEx);
                } else {
                    throw new RuntimeException("An unexpected error occurred while
decrypting data: " + exception.getMessage(), exception);
                }
            }
        });
```

```
        return responseFuture.thenApply(decryptResponse ->
decryptResponse.plaintext().asString(StandardCharsets.UTF_8));
    }
```

- 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.

```
    /**
     * Deletes a specific KMS alias asynchronously.
     *
     * @param aliasName the name of the alias to be deleted
     * @return a {@link CompletableFuture} representing the asynchronous operation
of deleting the specified alias
     */
    public CompletableFuture<Void> deleteSpecificAliasAsync(String aliasName) {
        DeleteAliasRequest deleteAliasRequest = DeleteAliasRequest.builder()
            .aliasName(aliasName)
            .build();

        return getAsyncClient().deleteAlias(deleteAliasRequest)
            .thenRun(() -> {
                logger.info("Alias {} has been deleted successfully", aliasName);
            })
            .exceptionally(throwable -> {
                throw new RuntimeException("Failed to delete alias: " + aliasName,
throwable);
            });
    }
```

- Per i dettagli sull'API, DeleteAliasconsulta 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.

```
    /**
     * Asynchronously checks if a specified key is enabled.
     *
     * @param keyId the ID of the key to check
     * @return a {@link CompletableFuture} that, when completed, indicates whether
the key is enabled or not
     *
     * @throws RuntimeException if an exception occurs while checking the key state
     */
    public CompletableFuture<Boolean> isKeyEnabledAsync(String keyId) {
        DescribeKeyRequest keyRequest = DescribeKeyRequest.builder()
            .keyId(keyId)
            .build();

        CompletableFuture<DescribeKeyResponse> responseFuture =
getAsyncClient().describeKey(keyRequest);
        return responseFuture.whenComplete((resp, ex) -> {
            if (resp != null) {
                KeyState keyState = resp.keyMetadata().keyState();
                if (keyState == KeyState.ENABLED) {
                    logger.info("The key is enabled.");
                } else {
                    logger.info("The key is not enabled. Key state: {}", keyState);
                }
            } else {
```

```
                throw new RuntimeException(ex);
            }
        }).thenApply(resp -> resp.keyMetadata().keyState() == KeyState.ENABLED);
    }
```

- Per i dettagli sull'API, DescribeKeyconsulta 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.

```
    /**
     * Asynchronously disables the specified AWS Key Management Service (KMS) key.
     *
     * @param keyId the ID or Amazon Resource Name (ARN) of the KMS key to be
disabled
     * @return a CompletableFuture that, when completed, indicates that the key has
been disabled successfully
     */
    public CompletableFuture<Void> disableKeyAsync(String keyId) {
        DisableKeyRequest keyRequest = DisableKeyRequest.builder()
            .keyId(keyId)
            .build();

        return getAsyncClient().disableKey(keyRequest)
            .thenRun(() -> {
                logger.info("Key {} has been disabled successfully",keyId);
            })
            .exceptionally(throwable -> {
                throw new RuntimeException("Failed to disable key: " + keyId,
throwable);
            });
```

```
    }
```

- Per i dettagli sull'API, consulta la DisableKeysezione 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.

```
   /**
    * Asynchronously enables the specified key.
    *
    * @param keyId the ID of the key to enable
    * @return a {@link CompletableFuture} that completes when the key has been
enabled
    */
   public CompletableFuture<Void> enableKeyAsync(String keyId) {
       EnableKeyRequest enableKeyRequest = EnableKeyRequest.builder()
           .keyId(keyId)
           .build();

       CompletableFuture<EnableKeyResponse> responseFuture =
getAsyncClient().enableKey(enableKeyRequest);
       responseFuture.whenComplete((response, exception) -> {
           if (exception == null) {
               logger.info("Key with ID [{}] has been enabled.", keyId);
           } else {
               if (exception instanceof KmsException kmsEx) {
                   throw new RuntimeException("KMS error occurred while enabling
key: " + kmsEx.getMessage(), kmsEx);
               } else {
                   throw new RuntimeException("An unexpected error occurred while
enabling key: " + exception.getMessage(), exception);
               }
```

```
            }
        });

        return responseFuture.thenApply(response -> null);
    }
```

- Per i dettagli sull'API, consulta la EnableKeysezione 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.

```
    /**
     * Encrypts the given text asynchronously using the specified KMS client and key
ID.
     *
     * @param keyId the ID of the KMS key to use for encryption
     * @param text the text to encrypt
     * @return a CompletableFuture that completes with the encrypted data as an
SdkBytes object
     */
    public CompletableFuture<SdkBytes> encryptDataAsync(String keyId, String text) {
        SdkBytes myBytes = SdkBytes.fromUtf8String(text);
        EncryptRequest encryptRequest = EncryptRequest.builder()
            .keyId(keyId)
            .plaintext(myBytes)
            .build();

        CompletableFuture<EncryptResponse> responseFuture =
getAsyncClient().encrypt(encryptRequest).toCompletableFuture();
        return responseFuture.whenComplete((response, ex) -> {
            if (response != null) {
                String algorithm = response.encryptionAlgorithm().toString();
```

```
                logger.info("The string was encrypted with algorithm {}.",
algorithm);
            } else {
                throw new RuntimeException(ex);
            }
        }).thenApply(EncryptResponse::ciphertextBlob);
    }
```

- 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.

```
    /**
     * Asynchronously lists all the aliases in the current AWS account.
     *
     * @return a {@link CompletableFuture} that completes when the list of aliases
has been processed
     */
    public CompletableFuture<Object> listAllAliasesAsync() {
        ListAliasesRequest aliasesRequest = ListAliasesRequest.builder()
            .limit(15)
            .build();

        ListAliasesPublisher paginator =
getAsyncClient().listAliasesPaginator(aliasesRequest);
        return paginator.subscribe(response -> {
                response.aliases().forEach(alias ->
                    logger.info("The alias name is: " + alias.aliasName())
                );
            })
            .thenApply(v -> null)
```

```
            .exceptionally(ex -> {
                if (ex.getCause() instanceof KmsException) {
                    KmsException e = (KmsException) ex.getCause();
                    throw new RuntimeException("A KMS exception occurred: " +
e.getMessage());
                } else {
                    throw new RuntimeException("An unexpected error occurred: " +
ex.getMessage());
                }
            });
    }
```

- Per i dettagli sull'API, consulta la [ListAliases](sezione 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.

```
    /**
     * Asynchronously displays the grant IDs for the specified key ID.
     *
     * @param keyId the ID of the AWS KMS key for which to list the grants
     * @return a {@link CompletableFuture} that, when completed, will be null if
the operation succeeded, or will throw a {@link RuntimeException} if the operation
failed
     * @throws RuntimeException if there was an error listing the grants, either due
to an {@link KmsException} or an unexpected error
     */
    public CompletableFuture<Object> displayGrantIdsAsync(String keyId) {
        ListGrantsRequest grantsRequest = ListGrantsRequest.builder()
            .keyId(keyId)
            .limit(15)
            .build();
```

```
        ListGrantsPublisher paginator =
getAsyncClient().listGrantsPaginator(grantsRequest);
        return paginator.subscribe(response -> {
                response.grants().forEach(grant -> {
                    logger.info("The grant Id is: " + grant.grantId());
                });
            })
            .thenApply(v -> null)
            .exceptionally(ex -> {
                Throwable cause = ex.getCause();
                if (cause instanceof KmsException) {
                    throw new RuntimeException("Failed to list grants: " +
cause.getMessage(), cause);
                } else {
                    throw new RuntimeException("An unexpected error occurred: " +
cause.getMessage(), cause);
                }
            });
    }
```

- Per i dettagli sull'API, consulta la ListGrantssezione 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.

```
    /**
     * Asynchronously retrieves the key policy for the specified key ID and policy
name.
     *
     * @param keyId       the ID of the AWS KMS key for which to retrieve the policy
     * @param policyName the name of the key policy to retrieve
```

```
     * @return a {@link CompletableFuture} that, when completed, contains the key
 policy as a {@link String}
     */
    public CompletableFuture<String> getKeyPolicyAsync(String keyId, String
 policyName) {
        GetKeyPolicyRequest policyRequest = GetKeyPolicyRequest.builder()
            .keyId(keyId)
            .policyName(policyName)
            .build();

        return getAsyncClient().getKeyPolicy(policyRequest)
            .thenApply(response -> {
                String policy = response.policy();
                logger.info("The response is: " + policy);
                return policy;
            })
            .exceptionally(ex -> {
                throw new RuntimeException("Failed to get key policy", ex);
            });
    }
```

- Per i dettagli sull'API, consulta la ListKeyPoliciessezione 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.services.kms.KmsAsyncClient;
import software.amazon.awssdk.services.kms.model.ListKeysRequest;
import software.amazon.awssdk.services.kms.paginators.ListKeysPublisher;
import java.util.concurrent.CompletableFuture;

/**
```

```
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * 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) {
        listAllKeys();
    }

    public static void listAllKeys() {
        KmsAsyncClient kmsAsyncClient = KmsAsyncClient.builder()
            .build();
        ListKeysRequest listKeysRequest = ListKeysRequest.builder()
            .limit(15)
            .build();

        /*
         * The `subscribe` method is required when using paginator methods in the
 AWS SDK
         * because paginator methods return an instance of a `ListKeysPublisher`,
 which is
         * based on a reactive stream. This allows asynchronous retrieval of
 paginated
         * results as they become available. By subscribing to the stream, we can
 process
         * each page of results as they are emitted.
         */
        ListKeysPublisher keysPublisher =
 kmsAsyncClient.listKeysPaginator(listKeysRequest);
        CompletableFuture<Void> future = keysPublisher
            .subscribe(r -> r.keys().forEach(key ->
                System.out.println("The key ARN is: " + key.keyArn() + ". The key Id
 is: " + key.keyId())))
            .whenComplete((result, exception) -> {
                if (exception != null) {
                    System.err.println("Error occurred: " + exception.getMessage());
                } else {
                    System.out.println("Successfully listed all keys.");
                }
            });
```

```
        try {
            future.join();
        } catch (Exception e) {
            System.err.println("Failed to list keys: " + e.getMessage());
        }
    }
}
```

- Per i dettagli sull'API, consulta la ListKeyssezione 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.

```
   /**
    * Revokes a grant for the specified AWS KMS key asynchronously.
    *
    * @param keyId   The ID or key ARN of the AWS KMS key.
    * @param grantId The identifier of the grant to be revoked.
    * @return A {@link CompletableFuture} representing the asynchronous operation
of revoking the grant.
    *         The {@link CompletableFuture} will complete with a {@link
RevokeGrantResponse} object
    *         if the operation is successful, or with a {@code null} value if an
error occurs.
    */
   public CompletableFuture<RevokeGrantResponse> revokeKeyGrantAsync(String keyId,
String grantId) {
       RevokeGrantRequest grantRequest = RevokeGrantRequest.builder()
           .keyId(keyId)
           .grantId(grantId)
           .build();
```

```
        CompletableFuture<RevokeGrantResponse> responseFuture =
getAsyncClient().revokeGrant(grantRequest);
        responseFuture.whenComplete((response, exception) -> {
            if (exception == null) {
                logger.info("Grant ID: [" + grantId + "] was successfully
revoked!");
            } else {
                if (exception instanceof KmsException kmsEx) {
                    if (kmsEx.getMessage().contains("Grant does not exist")) {
                        logger.info("The grant ID '" + grantId + "' does not exist.
Moving on...");
                    } else {
                        throw new RuntimeException("KMS error occurred: " +
kmsEx.getMessage(), kmsEx);
                    }
                } else {
                    throw new RuntimeException("An unexpected error occurred: " +
exception.getMessage(), exception);
                }
            }
        });

        return responseFuture;
    }
```

- Per i dettagli sull'API, consulta la RevokeGrantsezione 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.

```
    /**
```

```
    * Deletes a KMS key asynchronously.
    *
    * <p><strong>Warning:</strong> 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
becomes unrecoverable.
    * This means that any files, databases, or other data that were encrypted using
the deleted KMS key
    * will become permanently inaccessible. Exercise extreme caution when deleting
KMS keys.</p>
    *
    * @param keyId the ID of the KMS key to delete
    * @return a {@link CompletableFuture} that completes when the key deletion is
scheduled
    */
   public CompletableFuture<Void> deleteKeyAsync(String keyId) {
       ScheduleKeyDeletionRequest deletionRequest =
ScheduleKeyDeletionRequest.builder()
           .keyId(keyId)
           .pendingWindowInDays(7)
           .build();

       return getAsyncClient().scheduleKeyDeletion(deletionRequest)
           .thenRun(() -> {
               logger.info("Key {} will be deleted in 7 days", keyId);
           })
           .exceptionally(throwable -> {
               throw new RuntimeException("Failed to schedule key deletion for key
ID: " + keyId, throwable);
           });
   }
```

- Per i dettagli sull'API, consulta la ScheduleKeyDeletionsezione 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.

```
    /**
     * Asynchronously signs and verifies data using AWS KMS.
     *
     * <p>The method performs the following steps:
     * <ol>
     *     <li>Creates an AWS KMS key with the specified key spec, key usage, and
 origin.</li>
     *     <li>Signs the provided message using the created KMS key and the RSASSA-
PSS-SHA-256 algorithm.</li>
     *     <li>Verifies the signature of the message using the created KMS key and
 the RSASSA-PSS-SHA-256 algorithm.</li>
     * </ol>
     *
     * @return a {@link CompletableFuture} that completes with the result of the
 signature verification,
     *         {@code true} if the signature is valid, {@code false} otherwise.
     * @throws KmsException if any error occurs during the KMS operations.
     * @throws RuntimeException if an unexpected error occurs.
     */
    public CompletableFuture<Boolean> signVerifyDataAsync() {
        String signMessage = "Here is the message that will be digitally signed";

        // Create an AWS KMS key used to digitally sign data.
        CreateKeyRequest createKeyRequest = CreateKeyRequest.builder()
            .keySpec(KeySpec.RSA_2048)
            .keyUsage(KeyUsageType.SIGN_VERIFY)
            .origin(OriginType.AWS_KMS)
            .build();

        return getAsyncClient().createKey(createKeyRequest)
            .thenCompose(createKeyResponse -> {
                String keyId = createKeyResponse.keyMetadata().keyId();
```

```
                SdkBytes messageBytes = SdkBytes.fromString(signMessage,
Charset.defaultCharset());
                SignRequest signRequest = SignRequest.builder()
                    .keyId(keyId)
                    .message(messageBytes)
                    .signingAlgorithm(SigningAlgorithmSpec.RSASSA_PSS_SHA_256)
                    .build();

                return getAsyncClient().sign(signRequest)
                    .thenCompose(signResponse -> {
                        byte[] signedBytes = signResponse.signature().asByteArray();

                        VerifyRequest verifyRequest = VerifyRequest.builder()
                            .keyId(keyId)

.message(SdkBytes.fromByteArray(signMessage.getBytes(Charset.defaultCharset())))

.signature(SdkBytes.fromByteBuffer(ByteBuffer.wrap(signedBytes)))

.signingAlgorithm(SigningAlgorithmSpec.RSASSA_PSS_SHA_256)
                            .build();

                        return getAsyncClient().verify(verifyRequest)
                            .thenApply(verifyResponse -> {
                                return (boolean) verifyResponse.signatureValid();
                            });
                    });
            })
            .exceptionally(throwable -> {
               throw new RuntimeException("Failed to sign or verify data",
throwable);
            });
   }
```

- Per i dettagli sull'API, consulta Sign in AWS SDK for Java 2.x API Reference.

`TagResource`

Il seguente esempio di codice mostra come utilizzare`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.

```
    /**
     * Asynchronously tags a KMS key with a specific tag.
     *
     * @param keyId the ID of the KMS key to be tagged
     * @return a {@link CompletableFuture} that completes when the tagging operation
is finished
     */
    public CompletableFuture<Void> tagKMSKeyAsync(String keyId) {
        Tag tag = Tag.builder()
            .tagKey("Environment")
            .tagValue("Production")
            .build();

        TagResourceRequest tagResourceRequest = TagResourceRequest.builder()
            .keyId(keyId)
            .tags(tag)
            .build();

        return getAsyncClient().tagResource(tagResourceRequest)
            .thenRun(() -> {
                logger.info("{} key was tagged", keyId);
            })
            .exceptionally(throwable -> {
                throw new RuntimeException("Failed to tag the KMS key", throwable);
            });
    }
```

- Per i dettagli sull'API, consulta la TagResourcesezione AWS SDK for Java 2.x API Reference.

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 nozioni di base sono esempi di codice che mostrano come eseguire le operazioni essenziali all'interno di un servizio.

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.

Gli scenari sono esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio o combinate con altri Servizi AWS.

AWS i contributi della community sono esempi che sono stati creati e gestiti da più team AWS. Per fornire feedback, utilizza il meccanismo fornito negli archivi collegati.

Ogni esempio include un collegamento al codice sorgente completo, dove puoi 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.

```
    /**
     * Lists the AWS Lambda functions associated with the current AWS account.
     *
     * @param awsLambda an instance of the {@link LambdaClient} class, which is used
to interact with the AWS Lambda service
     *
```

```
     * @throws LambdaException if an error occurs while interacting with the AWS
Lambda service
     */
    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, consulta la ListFunctionssezione AWS SDK for Java 2.x API Reference.

Argomenti

- Nozioni di base
- Azioni
- Scenari
- Esempi serverless
- AWS contributi della comunità

Nozioni di base

Informazioni di base

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 altro su 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> <role> <handler> <bucketName> <key>\s

            Where:
                functionName - The name of the Lambda function.\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 != 5) {
              System.out.println(usage);
              return;
        }

        String functionName = args[0];
        String role = args[1];
        String handler = args[2];
        String bucketName = args[3];
        String key = args[4];
        LambdaClient awsLambda = LambdaClient.builder()
            .build();

        System.out.println(DASHES);
        System.out.println("Welcome to the AWS Lambda Basics scenario.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("1. Create an AWS Lambda function.");
        String funArn = createLambdaFunction(awsLambda, functionName, key,
bucketName, 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();
    }

    /**
     * Creates a new Lambda function in AWS using the AWS Lambda Java API.
     *
```

```
    * @param awsLambda    the AWS Lambda client used to interact with the AWS
Lambda service
    * @param functionName the name of the Lambda function to create
    * @param key          the S3 key of the function code
    * @param bucketName   the name of the S3 bucket containing the function code
    * @param role         the IAM role to assign to the Lambda function
    * @param handler      the fully qualified class name of the function handler
    * @return the Amazon Resource Name (ARN) of the created Lambda function
    */
   public static String createLambdaFunction(LambdaClient awsLambda,
                                              String functionName,
                                              String key,
                                              String bucketName,
                                              String role,
                                              String handler) {

       try {
           LambdaWaiter waiter = awsLambda.waiter();
           FunctionCode code = FunctionCode.builder()
               .s3Key(key)
               .s3Bucket(bucketName)
               .build();

           CreateFunctionRequest functionRequest = CreateFunctionRequest.builder()
               .functionName(functionName)
               .description("Created by the Lambda Java API")
               .code(code)
               .handler(handler)
               .runtime(Runtime.JAVA17)
               .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 e) {
```

```
            System.err.println(e.getMessage());
            System.exit(1);
        }
        return "";
    }

    /**
     * Retrieves information about an AWS Lambda function.
     *
     * @param awsLambda    an instance of the {@link LambdaClient} class, which is
used to interact with the AWS Lambda service
     * @param functionName the name of the AWS Lambda function to retrieve
information about
     */
    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);
        }
    }

    /**
     * Lists the AWS Lambda functions associated with the current AWS account.
     *
     * @param awsLambda an instance of the {@link LambdaClient} class, which is used
to interact with the AWS Lambda service
     *
     * @throws LambdaException if an error occurs while interacting with the AWS
Lambda service
     */
    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);
        }
    }

    /**
     * Invokes a specific AWS Lambda function.
     *
     * @param awsLambda    an instance of {@link LambdaClient} to interact with the
AWS Lambda service
     * @param functionName the name of the AWS Lambda function to be invoked
     */
    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);
        }
    }

    /**
     * Updates the code for an AWS Lambda function.
     *
     * @param awsLambda  the AWS Lambda client
```

```
    * @param functionName the name of the Lambda function to update
    * @param bucketName the name of the S3 bucket where the function code is
located
    * @param key the key (file name) of the function code in the S3 bucket
    * @throws LambdaException if there is an error updating the function code
    */
   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);
       }
   }

   /**
    * Updates the configuration of an AWS Lambda function.
    *
    * @param awsLambda     the {@link LambdaClient} instance to use for the AWS
Lambda operation
    * @param functionName  the name of the AWS Lambda function to update
    * @param handler       the new handler for the AWS Lambda function
```

```
    *
    * @throws LambdaException if there is an error while updating the function
configuration
    */
   public static void updateFunctionConfiguration(LambdaClient awsLambda, String
functionName, String handler) {
       try {
           UpdateFunctionConfigurationRequest configurationRequest =
UpdateFunctionConfigurationRequest.builder()
               .functionName(functionName)
               .handler(handler)
               .runtime(Runtime.JAVA17)
               .build();

           awsLambda.updateFunctionConfiguration(configurationRequest);

       } catch (LambdaException e) {
           System.err.println(e.getMessage());
           System.exit(1);
       }
   }

   /**
    * Deletes an AWS Lambda function.
    *
    * @param awsLambda     an instance of the {@link LambdaClient} class, which is
used to interact with the AWS Lambda service
    * @param functionName  the name of the Lambda function to be deleted
    *
    * @throws LambdaException if an error occurs while deleting the Lambda function
    */
   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

Azioni

`CreateFunction`

Il seguente esempio di codice mostra come usare`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.

```
    /**
     * Creates a new Lambda function in AWS using the AWS Lambda Java API.
     *
     * @param awsLambda    the AWS Lambda client used to interact with the AWS
Lambda service
     * @param functionName the name of the Lambda function to create
     * @param key          the S3 key of the function code
     * @param bucketName   the name of the S3 bucket containing the function code
     * @param role         the IAM role to assign to the Lambda function
```

```
     * @param handler      the fully qualified class name of the function handler
     * @return the Amazon Resource Name (ARN) of the created Lambda function
     */
    public static String createLambdaFunction(LambdaClient awsLambda,
                                               String functionName,
                                               String key,
                                               String bucketName,
                                               String role,
                                               String handler) {

        try {
            LambdaWaiter waiter = awsLambda.waiter();
            FunctionCode code = FunctionCode.builder()
                .s3Key(key)
                .s3Bucket(bucketName)
                .build();

            CreateFunctionRequest functionRequest = CreateFunctionRequest.builder()
                .functionName(functionName)
                .description("Created by the Lambda Java API")
                .code(code)
                .handler(handler)
                .runtime(Runtime.JAVA17)
                .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 e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        return "";
    }
```

- Per i dettagli sull'API, consulta la CreateFunctionsezione 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.

```
   /**
    * Deletes an AWS Lambda function.
    *
    * @param awsLambda     an instance of the {@link LambdaClient} class, which is
used to interact with the AWS Lambda service
    * @param functionName  the name of the Lambda function to be deleted
    *
    * @throws LambdaException if an error occurs while deleting the Lambda function
    */
   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, consulta la DeleteFunctionsezione AWS SDK for Java 2.x API Reference.

`GetFunction`

Il seguente esempio di codice mostra come utilizzare`GetFunction`.

SDK per Java 2.x

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
   /**
    * Retrieves information about an AWS Lambda function.
    *
    * @param awsLambda    an instance of the {@link LambdaClient} class, which is
used to interact with the AWS Lambda service
    * @param functionName the name of the AWS Lambda function to retrieve
information about
    */
   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);
       }
   }
```

- Per i dettagli sull'API, consulta la GetFunctionsezione 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.

```
    /**
     * Invokes a specific AWS Lambda function.
     *
     * @param awsLambda    an instance of {@link LambdaClient} to interact with the
AWS Lambda service
     * @param functionName the name of the AWS Lambda function to be invoked
     */
    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);
        }
    }
```

- Per informazioni dettagliate sulle API, consulta Invoke nella Documentazione di riferimento delle API AWS SDK for Java 2.x .

`UpdateFunctionCode`

Il seguente esempio di codice mostra come usare`UpdateFunctionCode`.

SDK per Java 2.x

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
   /**
    * Updates the code for an AWS Lambda function.
    *
    * @param awsLambda  the AWS Lambda client
    * @param functionName the name of the Lambda function to update
    * @param bucketName the name of the S3 bucket where the function code is
located
    * @param key the key (file name) of the function code in the S3 bucket
    * @throws LambdaException if there is an error updating the function code
    */
   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);
        }
    }
```

- Per i dettagli sull'API, consulta la UpdateFunctionCodesezione AWS SDK for Java 2.x API Reference.

`UpdateFunctionConfiguration`

Il seguente esempio di codice mostra come utilizzare`UpdateFunctionConfiguration`.

SDK per Java 2.x

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
    /**
     * Updates the configuration of an AWS Lambda function.
     *
     * @param awsLambda     the {@link LambdaClient} instance to use for the AWS
Lambda operation
     * @param functionName  the name of the AWS Lambda function to update
     * @param handler       the new handler for the AWS Lambda function
     *
```

```
    * @throws LambdaException if there is an error while updating the function
configuration
    */
   public static void updateFunctionConfiguration(LambdaClient awsLambda, String
functionName, String handler) {
       try {
           UpdateFunctionConfigurationRequest configurationRequest =
UpdateFunctionConfigurationRequest.builder()
               .functionName(functionName)
               .handler(handler)
               .runtime(Runtime.JAVA17)
               .build();

           awsLambda.updateFunctionConfiguration(configurationRequest);

       } catch (LambdaException e) {
           System.err.println(e.getMessage());
           System.exit(1);
       }
   }
```

- Per i dettagli sull'API, consulta la UpdateFunctionConfigurationsezione AWS SDK for Java 2.x API Reference.

Scenari

Creazione di un'applicazione serverless per gestire foto

Nell'esempio di codice seguente viene illustrato come creare un'applicazione serverless che consente agli utenti di gestire le foto mediante 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

Crea un'applicazione per analizzare il feedback dei clienti

L'esempio di codice seguente mostra come creare un'applicazione che analizza le schede dei commenti dei clienti, le traduce dalla loro lingua originale, ne determina il sentiment e genera un file audio dal testo tradotto.

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

Utilizzo di un'API Gateway per richiamare una funzione Lambda

Il seguente esempio di codice mostra come creare una AWS Lambda funzione richiamata da Amazon API Gateway.

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

Il seguente esempio di codice mostra come creare una macchina a AWS Step Functions stati che richiama AWS Lambda funzioni in sequenza.

SDK per Java 2.x

Mostra come creare un flusso di lavoro AWS senza server 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

Il seguente esempio di codice mostra come creare una AWS Lambda funzione richiamata da un evento EventBridge pianificato di Amazon.

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

- CloudWatch Registri
- DynamoDB
- EventBridge
- Lambda
- Amazon SNS

Usa l'API Neptune per interrogare i dati del grafico

Il seguente esempio di codice mostra come utilizzare l'API Neptune per interrogare i dati del grafico.

SDK per Java 2.x

Mostra come usare l'API Java di Amazon Neptune per creare una funzione Lambda che interroga i dati del grafico all'interno del VPC.

Per il codice sorgente completo e le istruzioni su come configurarlo ed eseguirlo, consulta l'esempio completo su. GitHub

Servizi utilizzati in questo esempio

- Lambda
- Neptune

Esempi serverless

Connessione a un database Amazon RDS in una funzione Lambda

Il seguente esempio di codice mostra come implementare una funzione Lambda che si connette a un database RDS. La funzione effettua una semplice richiesta al database e restituisce il risultato.

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.

Connessione a un database Amazon RDS in una funzione Lambda tramite Java.

```
import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.APIGatewayProxyRequestEvent;
import com.amazonaws.services.lambda.runtime.events.APIGatewayProxyResponseEvent;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rdsdata.RdsDataClient;
import software.amazon.awssdk.services.rdsdata.model.ExecuteStatementRequest;
import software.amazon.awssdk.services.rdsdata.model.ExecuteStatementResponse;
import software.amazon.awssdk.services.rdsdata.model.Field;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
```

```
public class RdsLambdaHandler implements RequestHandler<APIGatewayProxyRequestEvent,
 APIGatewayProxyResponseEvent> {

    @Override
    public APIGatewayProxyResponseEvent handleRequest(APIGatewayProxyRequestEvent
 event, Context context) {
        APIGatewayProxyResponseEvent response = new APIGatewayProxyResponseEvent();

        try {
            // Obtain auth token
            String token = createAuthToken();

            // Define connection configuration
            String connectionString = String.format("jdbc:mysql://%s:%s/%s?
useSSL=true&requireSSL=true",
                    System.getenv("ProxyHostName"),
                    System.getenv("Port"),
                    System.getenv("DBName"));

            // Establish a connection to the database
            try (Connection connection =
 DriverManager.getConnection(connectionString, System.getenv("DBUserName"), token);
                 PreparedStatement statement = connection.prepareStatement("SELECT ?
 + ? AS sum")) {

                statement.setInt(1, 3);
                statement.setInt(2, 2);

                try (ResultSet resultSet = statement.executeQuery()) {
                    if (resultSet.next()) {
                        int sum = resultSet.getInt("sum");
                        response.setStatusCode(200);
                        response.setBody("The selected sum is: " + sum);
                    }
                }
            }

        } catch (Exception e) {
            response.setStatusCode(500);
            response.setBody("Error: " + e.getMessage());
        }

        return response;
    }
```

```
    private String createAuthToken() {
        // Create RDS Data Service client
        RdsDataClient rdsDataClient = RdsDataClient.builder()
                .region(Region.of(System.getenv("AWS_REGION")))
                .credentialsProvider(DefaultCredentialsProvider.create())
                .build();

        // Define authentication request
        ExecuteStatementRequest request = ExecuteStatementRequest.builder()
                .resourceArn(System.getenv("ProxyHostName"))
                .secretArn(System.getenv("DBUserName"))
                .database(System.getenv("DBName"))
                .sql("SELECT 'RDS IAM Authentication'")
                .build();

        // Execute request and obtain authentication token
        ExecuteStatementResponse response = rdsDataClient.executeStatement(request);
        Field tokenField = response.records().get(0).get(0);

        return tokenField.stringValue();
    }
}
```

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;
    }

}
```

Richiamare una funzione Lambda da un trigger DynamoDB

Il seguente esempio di codice mostra come implementare una funzione Lambda che riceve un evento attivato dalla ricezione di record da un flusso DynamoDB. La funzione recupera il payload DocumentDB 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 DynamoDB con Lambda tramite Java.

```
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.DynamodbEvent.DynamodbStreamRecord;
import com.google.gson.Gson;
import com.google.gson.GsonBuilder;

public class example implements RequestHandler<DynamodbEvent, Void> {

    private static final Gson GSON = new GsonBuilder().setPrettyPrinting().create();

    @Override
    public Void handleRequest(DynamodbEvent event, Context context) {
        System.out.println(GSON.toJson(event));
        event.getRecords().forEach(this::logDynamoDBRecord);
        return null;
    }

    private void logDynamoDBRecord(DynamodbStreamRecord record) {
        System.out.println(record.getEventID());
        System.out.println(record.getEventName());
        System.out.println("DynamoDB Record: " + GSON.toJson(record.getDynamodb()));
    }
}
```

Richiamare una funzione Lambda da un trigger Amazon DocumentDB

Il seguente esempio di codice mostra come implementare una funzione Lambda che riceve un evento attivato dalla ricezione di record da un flusso di modifiche di DocumentDB. La funzione recupera il payload DocumentDB 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 Amazon DocumentDB con Lambda tramite Java.

```
import java.util.List;
import java.util.Map;

import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;

public class Example implements RequestHandler<Map<String, Object>, String> {

    @SuppressWarnings("unchecked")
    @Override
    public String handleRequest(Map<String, Object> event, Context context) {
        List<Map<String, Object>> events = (List<Map<String, Object>>)
 event.get("events");
        for (Map<String, Object> record : events) {
            Map<String, Object> eventData = (Map<String, Object>)
 record.get("event");
            processEventData(eventData);
        }

        return "OK";
    }

    @SuppressWarnings("unchecked")
    private void processEventData(Map<String, Object> eventData) {
        String operationType = (String) eventData.get("operationType");
        System.out.println("operationType: %s".formatted(operationType));

        Map<String, Object> ns = (Map<String, Object>) eventData.get("ns");

        String db = (String) ns.get("db");
        System.out.println("db: %s".formatted(db));
        String coll = (String) ns.get("coll");
        System.out.println("coll: %s".formatted(coll));
```

```
        Map<String, Object> fullDocument = (Map<String, Object>)
 eventData.get("fullDocument");
        System.out.println("fullDocument: %s".formatted(fullDocument));
    }

}
```

Invocare una funzione Lambda da un trigger Amazon MSK

Il seguente esempio di codice mostra come implementare una funzione Lambda che riceve un evento generato dalla ricezione di record da un cluster Amazon MSK. La funzione recupera il payload MSK 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 Amazon MSK con Lambda tramite Java.

```
import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.KafkaEvent;
import com.amazonaws.services.lambda.runtime.events.KafkaEvent.KafkaEventRecord;

import java.util.Base64;
import java.util.Map;

public class Example implements RequestHandler<KafkaEvent, Void> {

    @Override
    public Void handleRequest(KafkaEvent event, Context context) {
        for (Map.Entry<String, java.util.List<KafkaEventRecord>> entry :
 event.getRecords().entrySet()) {
            String key = entry.getKey();
```

```
            System.out.println("Key: " + key);

            for (KafkaEventRecord record : entry.getValue()) {
                System.out.println("Record: " + record);

                byte[] value = Base64.getDecoder().decode(record.getValue());
                String message = new String(value);
                System.out.println("Message: " + message);
            }
        }

        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.

Utilizzo di un evento SNS 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.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 di errori di elementi batch per 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 di errori di elementi batch di DynamoDB 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.DynamodbEvent;
import com.amazonaws.services.lambda.runtime.events.StreamsEventResponse;
import com.amazonaws.services.lambda.runtime.events.models.dynamodb.StreamRecord;

import java.util.ArrayList;
import java.util.List;

public class ProcessDynamodbRecords implements RequestHandler<DynamodbEvent,
 StreamsEventResponse> {

    @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
            } catch (Exception e) {
                //Add failed message identifier to the batchItemFailures list
                batchItemFailures.add(new
 SQSBatchResponse.BatchItemFailure(message.getMessageId()));
            }
        }
        return new SQSBatchResponse(batchItemFailures);
    }
}
```

AWS contributi della comunità

Crea e testa un'applicazione serverless

Il seguente esempio di codice mostra come creare e testare un'applicazione serverless utilizzando API Gateway con Lambda e DynamoDB

SDK per Java 2.x

> Mostra come creare e testare un'applicazione serverless composta da un API Gateway con Lambda e DynamoDB utilizzando Java SDK.
>
> Per il codice sorgente completo e le istruzioni su come configurarlo ed eseguirlo, consulta l'esempio completo su. GitHub
>
> Servizi utilizzati in questo esempio

- API Gateway

- DynamoDB
- Lambda

Esempi di Amazon Lex con SDK for Java 2.x

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando Amazon Lex. AWS SDK for Java 2.x

Gli scenari sono esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio o combinate con altri Servizi AWS.

Ogni esempio include un collegamento al codice sorgente completo, dove puoi trovare istruzioni su come configurare ed eseguire il codice nel contesto.

Argomenti

- Scenari

Scenari

Creazione di un chatbot Amazon Lex

Il seguente esempio di codice mostra come creare un chatbot 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

Esempi di Amazon Location 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 Location.

Le nozioni di base sono esempi di codice che mostrano come eseguire le operazioni essenziali all'interno di un servizio.

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.

Ogni esempio include un collegamento al codice sorgente completo, dove puoi trovare istruzioni su come configurare ed eseguire il codice nel contesto.

Nozioni di base

Salve Amazon Location Service

I seguenti esempi di codice mostrano come iniziare a utilizzare Amazon Location Service.

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
 *
 * In addition, you need to create a collection using the AWS Management
 * console. For information, see the following documentation.
 *
 * https://docs.aws.amazon.com/location/latest/developerguide/geofence-gs.html
```

```
 */
public class HelloLocation {

    private static LocationAsyncClient locationAsyncClient;
    private static final Logger logger =
 LoggerFactory.getLogger(HelloLocation.class);

    // This Singleton pattern ensures that only one `LocationClient`
    // instance.
    private static LocationAsyncClient getClient() {
        if (locationAsyncClient == null) {
            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                .maxConcurrency(100)
                .connectionTimeout(Duration.ofSeconds(60))
                .readTimeout(Duration.ofSeconds(60))
                .writeTimeout(Duration.ofSeconds(60))
                .build();

            ClientOverrideConfiguration overrideConfig =
 ClientOverrideConfiguration.builder()
                .apiCallTimeout(Duration.ofMinutes(2))
                .apiCallAttemptTimeout(Duration.ofSeconds(90))
                .retryStrategy(RetryMode.STANDARD)
                .build();

            locationAsyncClient = LocationAsyncClient.builder()
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return locationAsyncClient;
    }

    public static void main(String[] args) {
        final String usage = """

            Usage:
                <collectionName>

            Where:
                collectionName - The Amazon location collection name.
            """;

        if (args.length != 1) {
```

```
            System.out.println(usage);
            System.exit(1);
        }

        String collectionName = args[0];
        listGeofences(collectionName);
    }

    /**
     * Lists geofences from a specified geofence collection asynchronously.
     *
     * @param collectionName The name of the geofence collection to list geofences
 from.
     * @return A {@link CompletableFuture} representing the result of the
 asynchronous operation.
     *         The future completes when all geofences have been processed and
 logged.
     */
    public static CompletableFuture<Void> listGeofences(String collectionName) {
        ListGeofencesRequest geofencesRequest = ListGeofencesRequest.builder()
                .collectionName(collectionName)
                .build();

        ListGeofencesPublisher paginator =
 getClient().listGeofencesPaginator(geofencesRequest);
        CompletableFuture<Void> future = paginator.subscribe(response -> {
            if (response.entries().isEmpty()) {
                logger.info("No Geofences were found in the collection.");
            } else {
                response.entries().forEach(geofence ->
                        logger.info("Geofence ID: " + geofence.geofenceId())
                );
            }
        });
        return future;
    }
}
```

- Per i dettagli sull'API, consulta la [ListGeofencesPaginator](sezione AWS SDK for Java 2.x API Reference.

Argomenti

- Nozioni di base
- Azioni

Nozioni di base

Informazioni di base

L'esempio di codice seguente mostra come:

- Crea una mappa delle sedi di Amazon.
- Crea una chiave API Amazon Location.
- Visualizza l'URL della mappa.
- Crea una raccolta di geofence.
- Memorizza una geometria di geofence.
- Crea una risorsa di tracciamento.
- Aggiorna la posizione di un dispositivo.
- Recupera l'aggiornamento di posizione più recente per un dispositivo specificato.
- Crea un calcolatore di percorso.
- Determina la distanza tra Seattle e Vancouver.
- Usa il livello superiore di Amazon Location APIs.
- Elimina le risorse di localizzazione di Amazon.

SDK per Java 2.x

> **Note**
>
> C'è altro da fare 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 Amazon Location Service.

```
/*
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
```

```
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 */
public class LocationScenario {

    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    private static final Logger logger =
 LoggerFactory.getLogger(LocationScenario.class);
    static Scanner scanner = new Scanner(System.in);

    static LocationActions locationActions = new LocationActions();

    public static void main(String[] args) {
        final String usage = """

            Usage:    <mapName> <keyName> <collectionName> <geoId> <trackerName>
 <calculatorName> <deviceId>

            Where:
              mapName - The name of the map to be create (e.g., "AWSMap").
              keyName - The name of the API key to create (e.g., "AWSApiKey").
              collectionName - The name of the geofence collection (e.g.,
 "AWSLocationCollection").
              geoId - The geographic identifier used for the geofence or map (e.g.,
 "geoId").
              trackerName - The name of the tracker (e.g., "geoTracker").
              calculatorName - The name of the route calculator (e.g.,
 "AWSRouteCalc").
              deviceId - The ID of the device (e.g., "iPhone-112356").
            """;

        if (args.length != 7) {
            logger.info(usage);
            return;
        }

        String mapName = args[0];
        String keyName = args[1];
        String collectionName = args[2];
        String geoId = args[3];
        String trackerName = args[4];
```

```
        String calculatorName = args[5];
        String deviceId = args[6];

        logger.info("""
            AWS Location Service is a fully managed service offered by Amazon Web
Services (AWS) that
            provides location-based services for developers. This service simplifies
            the integration of location-based features into applications, making it
            easier to build and deploy location-aware applications.

            The AWS Location Service offers a range of location-based services,
            including:

            Maps: The service provides access to high-quality maps, satellite
imagery,\s
            and geospatial data from various providers, allowing developers to\s
            easily embed maps into their applications.

            Tracking: The Location Service enables real-time tracking of mobile
devices,\s
            assets, or other entities, allowing developers to build applications\s
            that can monitor the location of people, vehicles, or other objects.

            Geocoding: The service provides the ability to convert addresses or\s
            location names into geographic coordinates (latitude and longitude),\s
            and vice versa, enabling developers to integrate location-based search\s
            and routing functionality into their applications.
            """);
        waitForInputToContinue(scanner);
        try {
            runScenario(mapName, keyName, collectionName, geoId, trackerName,
calculatorName, deviceId);
        } catch (RuntimeException e) {
            // Clean up AWS Resources.
            cleanUp(mapName, keyName, collectionName, trackerName, calculatorName);
            logger.info(e.getMessage());
        }
    }

    public static void runScenario(String mapName, String keyName, String
collectionName, String geoId, String trackerName, String calculatorName, String
deviceId) {
        logger.info(DASHES);
        logger.info("1. Create a map");
```

```
        logger.info("""
             An AWS Location map can enhance the user experience of your
             application by providing accurate and personalized location-based
             features. For example, you could use the geocoding capabilities to
             allow users to search for and locate businesses, landmarks, or
             other points of interest within a specific region.
            """);

        waitForInputToContinue(scanner);
        String mapArn;
        try {
            mapArn = locationActions.createMap(mapName).join();
            logger.info("The Map ARN is: {}", mapArn);  // Log success in calling
code
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ServiceQuotaExceededException) {
                logger.error("The request exceeded the maximum quota: {}",
cause.getMessage());
            } else {
                logger.error("An unexpected error occurred while creating the map.",
cause);
            }
            return;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("2. Create an AWS Location API key");
        logger.info("""
            When you embed a map in a web app or website, the API key is
            included in the map tile URL to authenticate requests. You can
            restrict API keys to specific AWS Location operations (e.g., only
            maps, not geocoding). API keys can expire, ensuring temporary
            access control.
            """);

        try {
            String keyArn = locationActions.createKey(keyName, mapArn).join();
            logger.info("The API key was successfully created: {}", keyArn);
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof AccessDeniedException) {
```

```
                logger.error("Request was denied: {}", cause.getMessage());
            } else {
                logger.error("An unexpected error occurred while creating the API
 key.", cause);
            }
            return;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("3. Display Map URL");
        logger.info("""
            In order to get the MAP URL, you need to get the API Key value.
            You can get the key value using the AWS Management Console under
            Location Services. This operation cannot be completed using the
            AWS SDK. For more information about getting the key value, see
            the AWS Location Documentation.
            """);
        String mapUrl = "https://maps.geo.aws.amazon.com/maps/v0/maps/"+mapName+"/
tiles/{z}/{x}/{y}?key={KeyValue}";
        logger.info("Embed this URL in your Web app: " + mapUrl);
        logger.info("");
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("4. Create a geofence collection, which manages and stores
 geofences.");
        waitForInputToContinue(scanner);
        try {
            String collectionArn =
 locationActions.createGeofenceCollection(collectionName).join();
            logger.info("The geofence collection was successfully created: {}",
 collectionArn);
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ConflictException) {
                logger.error("A conflict occurred: {}", cause.getMessage());
            } else {
                logger.error("An unexpected error occurred while creating the
 geofence collection.", cause);
            }
            return;
```

```
        }

        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("5. Store a geofence geometry in a given geofence collection.");
        logger.info("""
            An AWS Location geofence is a virtual boundary that defines a geographic
area
            on a map. It is a useful feature for tracking the location of
            assets or monitoring the movement of objects within a specific region.

            To define a geofence, you need to specify the coordinates of a
            polygon that represents the area of interest. The polygon must be
            defined in a counter-clockwise direction, meaning that the points of
            the polygon must be listed in a counter-clockwise order.

            This is a requirement for the AWS Location service to correctly
            interpret the geofence and ensure that the location data is
            accurately processed within the defined area.
            """);

        waitForInputToContinue(scanner);
        try {
            locationActions.putGeofence(collectionName, geoId).join();
            logger.info("Successfully created geofence: {}", geoId);
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ValidationException) {
                logger.error("A validation error occurred while creating geofence:
{}", cause.getMessage());
            } else {
                logger.error("An unexpected error occurred: {}", cause.getMessage(),
cause);
            }
            return;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info("6. Create a tracker resource which lets you retrieve current
and historical location of devices..");
        waitForInputToContinue(scanner);
```

```
        try {
            String trackerArn = locationActions.createTracker(trackerName).join();
            logger.info("Successfully created tracker. ARN: {}", trackerArn);  //
Log success
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ConflictException) {
                logger.error("A conflict occurred while creating the tracker: {}",
cause.getMessage());
            } else {
                logger.error("An unexpected error occurred: {}", cause.getMessage(),
cause);
            }
            return;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("7. Update the position of a device in the location tracking
system.");
        logger.info("""
            The AWS location service does not enforce a strict format for deviceId,
but it must:
              - Be a string (case-sensitive).
              - Be 1-100 characters long.
              - Contain only:
                - Alphanumeric characters (A-Z, a-z, 0-9)
                - Underscores (_)
                - Hyphens (-)
                - Be the same ID used when sending and retrieving positions.
            """);

        waitForInputToContinue(scanner);
        try {
            CompletableFuture<BatchUpdateDevicePositionResponse> future =
locationActions.updateDevicePosition(trackerName, deviceId);
            future.join();
            logger.info(deviceId + " was successfully updated in the location
tracking system.");
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ResourceNotFoundException) {
```

```
                logger.info("The resource was not found: {}", cause.getMessage(),
cause);
            } else {
                logger.info("An unexpected error occurred: {}", cause.getMessage(),
cause);
            }
            return;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info("8. Retrieve the most recent position update for a specified
device..");
        waitForInputToContinue(scanner);
        try {
            GetDevicePositionResponse response =
locationActions.getDevicePosition(trackerName, deviceId).join();
            logger.info("Successfully fetched device position: {}",
response.position());
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ResourceNotFoundException) {
                logger.info("The resource was not found: {}", cause.getMessage(),
cause);
            } else {
                logger.info("An unexpected error occurred: {}", cause.getMessage(),
cause);
            }
            return;
        }

        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info("9. Create a route calculator.");
        waitForInputToContinue(scanner);
        try {
            CreateRouteCalculatorResponse response =
locationActions.createRouteCalculator(calculatorName).join();
            logger.info("Route calculator created successfully: {}",
response.calculatorArn());
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ConflictException) {
```

```
                logger.info("A conflict occurred: {}", cause.getMessage(), cause);
            } else {
                logger.info("An unexpected error occurred: {}", cause.getMessage(),
cause);
            }
            return;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info("10. Determine the distance between Seattle and Vancouver using
the route calculator.");
        waitForInputToContinue(scanner);
        try {
            CalculateRouteResponse response =
locationActions.calcDistanceAsync(calculatorName).join();
            logger.info("Successfully calculated route. The distance in kilometers
is {}", response.summary().distance());
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ResourceNotFoundException) {
                logger.info("The resource was not found: {}", cause.getMessage(),
cause);
            } else {
                logger.info("An unexpected error occurred: {}", cause.getMessage(),
cause);
            }
            return;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info("11. Use the GeoPlacesAsyncClient to perform additional
operations.");
        logger.info("""
            This scenario will show use of the GeoPlacesClient that enables
            location search and geocoding capabilities for your applications.\s

            We are going to use this client to perform these AWS Location tasks:
             - Reverse Geocoding (reverseGeocode): Converts geographic coordinates
into addresses.
             - Place Search (searchText): Finds places based on search queries.
             - Nearby Search (searchNearby): Finds places near a specific location.
            """);
```

```
        logger.info("First we will perform a Reverse Geocoding operation");
        waitForInputToContinue(scanner);
        try {
            locationActions.reverseGeocode().join();
            logger.info("Now we are going to perform a text search using coffee
shop.");
            waitForInputToContinue(scanner);
            locationActions.searchText("coffee shop").join();
            waitForInputToContinue(scanner);

            logger.info("Now we are going to perform a nearby Search.");
            //waitForInputToContinue(scanner);
            locationActions.searchNearBy().join();
            waitForInputToContinue(scanner);
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof
software.amazon.awssdk.services.geoplaces.model.ValidationException) {
                logger.error("A validation error occurred: {}", cause.getMessage(),
cause);
            } else {
                logger.error("An unexpected error occurred: {}", cause.getMessage(),
cause);
            }
            return;
        }
        logger.info(DASHES);

        logger.info("12. Delete the AWS Location Services resources.");
        logger.info("Would you like to delete the AWS Location Services resources?
(y/n)");
        String delAns = scanner.nextLine().trim();
        if (delAns.equalsIgnoreCase("y")) {
            cleanUp(mapName, keyName, collectionName, trackerName, calculatorName);
        } else {
            logger.info("The AWS resources will not be deleted.");
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info(" This concludes the AWS Location Service scenario.");
        logger.info(DASHES);
```

```
    }

    /**
     * Cleans up resources by deleting the specified map, key, geofence collection,
tracker, and route calculator.
     *
     * @param mapName The name of the map to delete.
     * @param keyName The name of the key to delete.
     * @param collectionName The name of the geofence collection to delete.
     * @param trackerName The name of the tracker to delete.
     * @param calculatorName The name of the route calculator to delete.
     */
    private static void cleanUp(String mapName, String keyName, String
collectionName, String trackerName, String calculatorName) {
        try {
            locationActions.deleteMap(mapName).join();
            locationActions.deleteKey(keyName).join();
            locationActions.deleteGeofenceCollectionAsync(collectionName).join();
            locationActions.deleteTracker(trackerName).join();
            locationActions.deleteRouteCalculator(calculatorName).join();
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ResourceNotFoundException) {
                logger.info("The resource was not found: {}", cause.getMessage(),
cause);
            } else {
                logger.info("An unexpected error occurred: {}", cause.getMessage(),
cause);
            }
            return;
        }
    }

    private static void waitForInputToContinue(Scanner scanner) {
        while (true) {
            logger.info("");
            logger.info("Enter 'c' followed by <ENTER> to continue:");
            String input = scanner.nextLine();

            if (input.trim().equalsIgnoreCase("c")) {
                logger.info("Continuing with the program...");
                logger.info("");
                break;
            } else {
```

```
                logger.info("Invalid input. Please try again.");
            }
        }
    }
}
```

Una classe wrapper per i metodi SDK di Amazon Location Service.

```
public class LocationActions {

    private static LocationAsyncClient locationAsyncClient;

    private static GeoPlacesAsyncClient geoPlacesAsyncClient;
    private static final Logger logger =
 LoggerFactory.getLogger(LocationActions.class);

    // This Singleton pattern ensures that only one `LocationClient`
    // instance is used throughout the application.
    private LocationAsyncClient getClient() {
        if (locationAsyncClient == null) {
            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                .maxConcurrency(100)
                .connectionTimeout(Duration.ofSeconds(60))
                .readTimeout(Duration.ofSeconds(60))
                .writeTimeout(Duration.ofSeconds(60))
                .build();

            ClientOverrideConfiguration overrideConfig =
 ClientOverrideConfiguration.builder()
                .apiCallTimeout(Duration.ofMinutes(2))
                .apiCallAttemptTimeout(Duration.ofSeconds(90))
                .retryStrategy(RetryMode.STANDARD)
                .build();

            locationAsyncClient = LocationAsyncClient.builder()
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return locationAsyncClient;
    }
```

```
    private static GeoPlacesAsyncClient getGeoPlacesClient() {
        if (geoPlacesAsyncClient == null) {
            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                .maxConcurrency(100)
                .connectionTimeout(Duration.ofSeconds(60))
                .readTimeout(Duration.ofSeconds(60))
                .writeTimeout(Duration.ofSeconds(60))
                .build();

            ClientOverrideConfiguration overrideConfig =
 ClientOverrideConfiguration.builder()
                .apiCallTimeout(Duration.ofMinutes(2))
                .apiCallAttemptTimeout(Duration.ofSeconds(90))
                .retryStrategy(RetryMode.STANDARD)
                .build();

            geoPlacesAsyncClient = GeoPlacesAsyncClient.builder()
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return geoPlacesAsyncClient;
    }


    /**
     * Performs a nearby places search based on the provided geographic coordinates
 (latitude and longitude).
     * The method sends an asynchronous request to search for places within a 1-
kilometer radius of the specified location.
     * The results are processed and printed once the search completes successfully.
     */
    public CompletableFuture<SearchNearbyResponse> searchNearBy() {
        double latitude = 37.7749;  // San Francisco
        double longitude = -122.4194;
        List<Double> queryPosition = List.of(longitude, latitude);

        // Set up the request for searching nearby places.
        SearchNearbyRequest request = SearchNearbyRequest.builder()
            .queryPosition(queryPosition)  // Set the position
            .queryRadius(1000L)  // Radius in meters (1000 meters = 1 km).
            .build();

        return getGeoPlacesClient().searchNearby(request)
```

```
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof
software.amazon.awssdk.services.geoplaces.model.ValidationException) {
                        throw new CompletionException("A validation error occurred:
" + cause.getMessage(), cause);
                    }
                    throw new CompletionException("Error performing place search",
exception);
                }

                // Process the response and print the results.
                response.resultItems().forEach(result -> {
                    logger.info("Place Name: " + result.placeType().name());
                    logger.info("Address: " + result.address().label());
                    logger.info("Distance: " + result.distance() + " meters");
                    logger.info("-------------------------");
                });
            });
    }


    /**
     * Searches for a place using the provided search query and prints the detailed
information of the first result.
     *
     * @param searchQuery the search query to be used for the place search (ex,
coffee shop)
     */
    public CompletableFuture<Void> searchText(String searchQuery) {
        double latitude = 37.7749;  // San Francisco
        double longitude = -122.4194;
        List<Double> queryPosition = List.of(longitude, latitude);

        SearchTextRequest request = SearchTextRequest.builder()
                .queryText(searchQuery)
                .biasPosition(queryPosition)
                .build();

        return getGeoPlacesClient().searchText(request)
                .thenCompose(response -> {
                    if (response.resultItems().isEmpty()) {
                        logger.info("No places found.");
```

```
                        return CompletableFuture.completedFuture(null);
                    }

                    // Get the first place ID
                    String placeId = response.resultItems().get(0).placeId();
                    logger.info("Found Place with id: " + placeId);

                    // Fetch detailed info using getPlace
                    GetPlaceRequest getPlaceRequest = GetPlaceRequest.builder()
                            .placeId(placeId)
                            .build();

                    return getGeoPlacesClient().getPlace(getPlaceRequest)
                            .thenAccept(placeResponse -> {
                                logger.info("Detailed Place Information:");
                                logger.info("Name: " +
 placeResponse.placeType().name());
                                logger.info("Address: " +
 placeResponse.address().label());

                                if (placeResponse.foodTypes() != null && !
placeResponse.foodTypes().isEmpty()) {
                                    logger.info("Food Types:");
                                    placeResponse.foodTypes().forEach(foodType -> {
                                        logger.info("  - " + foodType);
                                    });
                                } else {
                                    logger.info("No food types available.");
                                }
                                logger.info("------------------------");
                            });
                })
                .exceptionally(exception -> {
                    Throwable cause = exception.getCause();
                    if (cause instanceof
 software.amazon.awssdk.services.geoplaces.model.ValidationException) {
                        throw new CompletionException("A validation error occurred:
 " + cause.getMessage(), cause);
                    }
                    throw new CompletionException("Error performing place search",
 exception);
                });
    }
```

```
   /**
    * Performs reverse geocoding using the AWS Geo Places API.
    * Reverse geocoding is the process of converting geographic coordinates
(latitude and longitude) to a human-readable address.
    * This method uses the latitude and longitude of San Francisco as the input,
and prints the resulting address.
    */
   public CompletableFuture<ReverseGeocodeResponse> reverseGeocode() {
       double latitude = 37.7749;  // San Francisco
       double longitude = -122.4194;
       logger.info("Use latitude 37.7749 and longitude -122.4194");

       // AWS expects [longitude, latitude].
       List<Double> queryPosition = List.of(longitude, latitude);
       ReverseGeocodeRequest request = ReverseGeocodeRequest.builder()
           .queryPosition(queryPosition)
           .build();
       CompletableFuture<ReverseGeocodeResponse> futureResponse =
           getGeoPlacesClient().reverseGeocode(request);

       return futureResponse.whenComplete((response, exception) -> {
           if (exception != null) {
               Throwable cause = exception.getCause();
               if (cause instanceof
software.amazon.awssdk.services.geoplaces.model.ValidationException) {
                   throw new CompletionException("A validation error occurred: " +
cause.getMessage(), cause);
               }
               throw new CompletionException("Error performing reverse geocoding",
exception);
           }

           response.resultItems().forEach(result ->
               logger.info("The address is: " + result.address().label())
           );
       });
   }


   /**
    * Calculates the distance between two locations asynchronously.
    *
```

```
    * @param routeCalcName the name of the route calculator to use
    * @return a {@link CompletableFuture} that will complete with a {@link
CalculateRouteResponse} containing the distance and estimated duration of the route
    */
   public CompletableFuture<CalculateRouteResponse> calcDistanceAsync(String
routeCalcName) {
       // Define coordinates for Seattle, WA and Vancouver, BC.
       List<Double> departurePosition = Arrays.asList(-122.3321, 47.6062);
       List<Double> arrivePosition = Arrays.asList(-123.1216, 49.2827);

       CalculateRouteRequest request = CalculateRouteRequest.builder()
           .calculatorName(routeCalcName)
           .departurePosition(departurePosition)
           .destinationPosition(arrivePosition)
           .travelMode("Car") // Options: Car, Truck, Walking, Bicycle
           .distanceUnit("Kilometers") // Options: Meters, Kilometers, Miles
           .build();

       return getClient().calculateRoute(request)
           .whenComplete((response, exception) -> {
               if (exception != null) {
                   Throwable cause = exception.getCause();
                   if (cause instanceof ResourceNotFoundException) {
                       throw new CompletionException("The AWS resource was not
found: " + cause.getMessage(), cause);
                   }
                   throw new CompletionException("Failed to calculate route: " +
exception.getMessage(), exception);
               }
           });
   }


   /**
    * Creates a new route calculator with the specified name and data source.
    *
    * @param routeCalcName the name of the route calculator to be created
    */
   public CompletableFuture<CreateRouteCalculatorResponse>
createRouteCalculator(String routeCalcName) {
       String dataSource = "Esri"; // or "Here"
       CreateRouteCalculatorRequest request =
CreateRouteCalculatorRequest.builder()
           .calculatorName(routeCalcName)
```

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

        return getClient().createRouteCalculator(request)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof ConflictException) {
                        throw new CompletionException("A conflict error occurred: "
+ cause.getMessage(), cause);
                    }
                    throw new CompletionException("Failed to create route
calculator: " + exception.getMessage(), exception);
                }
            });
    }


    /**
     * Retrieves the position of a device using the provided LocationClient.
     *
     * @param trackerName The name of the tracker associated with the device.
     * @param deviceId    The ID of the device to retrieve the position for.
     * @throws RuntimeException If there is an error fetching the device position.
     */
    public CompletableFuture<GetDevicePositionResponse> getDevicePosition(String
trackerName, String deviceId) {
        GetDevicePositionRequest request = GetDevicePositionRequest.builder()
            .trackerName(trackerName)
            .deviceId(deviceId)
            .build();

        return getClient().getDevicePosition(request)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof ResourceNotFoundException) {
                        throw new CompletionException("The AWS resource was not
found: " + cause.getMessage(), cause);
                    }
                    throw new CompletionException("Error fetching device position: "
+ exception.getMessage(), exception);
                }
            });
```

```
    }


    /**
     * Updates the position of a device in the location tracking system.
     *
     * @param trackerName the name of the tracker associated with the device
     * @param deviceId    the unique identifier of the device
     * @throws RuntimeException if an error occurs while updating the device
position
     */
    public CompletableFuture<BatchUpdateDevicePositionResponse>
updateDevicePosition(String trackerName, String deviceId) {
        double latitude = 37.7749;  // Example: San Francisco
        double longitude = -122.4194;

        DevicePositionUpdate positionUpdate = DevicePositionUpdate.builder()
            .deviceId(deviceId)
            .sampleTime(Instant.now()) // Timestamp of position update.
            .position(Arrays.asList(longitude, latitude)) // AWS requires
[longitude, latitude]
            .build();

        BatchUpdateDevicePositionRequest request =
BatchUpdateDevicePositionRequest.builder()
            .trackerName(trackerName)
            .updates(positionUpdate)
            .build();

        CompletableFuture<BatchUpdateDevicePositionResponse> futureResponse =
getClient().batchUpdateDevicePosition(request);
        return futureResponse.whenComplete((response, exception) -> {
            if (exception != null) {
                Throwable cause = exception.getCause();
                if (cause instanceof ResourceNotFoundException) {
                    throw new CompletionException("The resource was not found: " +
cause.getMessage(), cause);
                } else {
                    throw new CompletionException("Error updating device position: "
+ exception.getMessage(), exception);
                }
            }
        });
    }
```

```
   /**
    * Creates a new tracker resource in your AWS account, which you can use to
track the location of devices.
    *
    * @param trackerName the name of the tracker to be created
    * @return a {@link CompletableFuture} that, when completed, will contain the
Amazon Resource Name (ARN) of the created tracker
    */
   public CompletableFuture<String> createTracker(String trackerName) {
       CreateTrackerRequest trackerRequest = CreateTrackerRequest.builder()
           .description("Created using the Java V2 SDK")
           .trackerName(trackerName)
           .positionFiltering("TimeBased") // Options: TimeBased, DistanceBased,
AccuracyBased
           .build();

       return getClient().createTracker(trackerRequest)
           .whenComplete((response, exception) -> {
               if (exception != null) {
                   Throwable cause = exception.getCause();
                   if (cause instanceof ConflictException) {
                       throw new CompletionException("Conflict occurred while
creating tracker: " + cause.getMessage(), cause);
                   }
                   throw new CompletionException("Error creating tracker: " +
exception.getMessage(), exception);
               }
           })
           .thenApply(CreateTrackerResponse::trackerArn); // Return only the
tracker ARN
   }



   /**
    * Adds a new geofence to the specified collection.
    *
    * @param collectionName the name of the geofence collection to add the geofence
to
    * @param geoId          the unique identifier for the geofence
    */
```

```
   public CompletableFuture<PutGeofenceResponse> putGeofence(String collectionName,
String geoId) {
       // Define the geofence geometry (polygon).
       GeofenceGeometry geofenceGeometry = GeofenceGeometry.builder()
           .polygon(List.of(
               List.of(
                   List.of(-122.3381, 47.6101), // First point
                   List.of(-122.3281, 47.6101),
                   List.of(-122.3281, 47.6201),
                   List.of(-122.3381, 47.6201),
                   List.of(-122.3381, 47.6101) // Closing the polygon
               )
           ))
           .build();

       PutGeofenceRequest geofenceRequest = PutGeofenceRequest.builder()
           .collectionName(collectionName) // Specify the collection.
           .geofenceId(geoId) // Unique ID for the geofence.
           .geometry(geofenceGeometry)
           .build();

       return getClient().putGeofence(geofenceRequest)
           .whenComplete((response, exception) -> {
               if (exception != null) {
                   Throwable cause = exception.getCause();
                   if (cause instanceof ValidationException) {
                       throw new CompletionException("Validation error while
creating geofence: " + cause.getMessage(), cause);
                   }
                   throw new CompletionException("Error creating geofence: " +
exception.getMessage(), exception);
               }
           });
   }


   /**
    * Creates a new geofence collection.
    *
    * @param collectionName the name of the geofence collection to be created
    */
   public CompletableFuture<String> createGeofenceCollection(String collectionName)
{
```

```
        CreateGeofenceCollectionRequest collectionRequest =
CreateGeofenceCollectionRequest.builder()
            .collectionName(collectionName)
            .description("Created by using the AWS SDK for Java")
            .build();

        return getClient().createGeofenceCollection(collectionRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof ConflictException) {
                        throw new CompletionException("The geofence collection was
not created due to ConflictException.", cause);
                    }
                    throw new CompletionException("Failed to create geofence
collection: " + exception.getMessage(), exception);
                }
            })
            .thenApply(response -> response.collectionArn()); // Return only the ARN
    }




    /**
     * Creates a new API key with the specified name and restrictions.
     *
     * @param keyName the name of the API key to be created
     * @param mapArn  the Amazon Resource Name (ARN) of the map resource to which
the API key will be associated
     * @return a {@link CompletableFuture} that completes with the Amazon Resource
Name (ARN) of the created API key,
     * or {@code null} if the operation failed
     */
    public CompletableFuture<String> createKey(String keyName, String mapArn) {
        ApiKeyRestrictions keyRestrictions = ApiKeyRestrictions.builder()
            .allowActions("geo:GetMap*")
            .allowResources(mapArn)
            .build();

        CreateKeyRequest request = CreateKeyRequest.builder()
            .keyName(keyName)
            .restrictions(keyRestrictions)
            .noExpiry(true)
```

```
            .build();

        return getClient().createKey(request)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof AccessDeniedException) {
                        throw new CompletionException("The request was denied
because of insufficient access or permissions.", cause);
                    }
                    throw new CompletionException("Failed to create API key: " +
exception.getMessage(), exception);
                }
            })
            .thenApply(response -> response.keyArn()); // This will never return
null if the response reaches here
    }



    /**
     * Creates a new map with the specified name and configuration.
     *
     * @param mapName the name of the map to be created
     * @return a {@link CompletableFuture} that, when completed, will contain the
Amazon Resource Name (ARN) of the created map
     * @throws CompletionException if an error occurs while creating the map, such
as exceeding the service quota
     */
    public CompletableFuture<String> createMap(String mapName) {
        MapConfiguration configuration = MapConfiguration.builder()
            .style("VectorEsriNavigation")
            .build();

        CreateMapRequest mapRequest = CreateMapRequest.builder()
            .mapName(mapName)
            .configuration(configuration)
            .description("A map created using the Java V2 API")
            .build();

        return getClient().createMap(mapRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
```

```
                    if (cause instanceof ServiceQuotaExceededException) {
                        throw new CompletionException("The operation was denied
because the request would exceed the maximum quota.", cause);
                    }
                    throw new CompletionException("Failed to create map: " +
exception.getMessage(), exception);
                }
            })
            .thenApply(response -> response.mapArn()); // Return the map ARN
    }



    /**
     * Deletes a geofence collection asynchronously.
     *
     * @param collectionName the name of the geofence collection to be deleted
     * @return a {@link CompletableFuture} that completes when the geofence
collection has been deleted
     */
    public CompletableFuture<Void> deleteGeofenceCollectionAsync(String
collectionName) {
        DeleteGeofenceCollectionRequest collectionRequest =
DeleteGeofenceCollectionRequest.builder()
            .collectionName(collectionName)
            .build();

        return getClient().deleteGeofenceCollection(collectionRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof ResourceNotFoundException) {
                        throw new CompletionException("The requested geofence
collection was not found.", cause);
                    }
                    throw new CompletionException("Failed to delete geofence
collection: " + exception.getMessage(), exception);
                }
                logger.info("The geofence collection {} was deleted.",
collectionName);
            })
            .thenApply(response -> null);
    }
```

```
   /**
    * Deletes the specified key from the key-value store.
    *
    * @param keyName the name of the key to be deleted
    * @return a {@link CompletableFuture} that completes when the key has been
deleted
    * @throws CompletionException if the key was not found or if an error occurred
during the deletion process
    */
   public CompletableFuture<Void> deleteKey(String keyName) {
       DeleteKeyRequest keyRequest = DeleteKeyRequest.builder()
           .keyName(keyName)
           .build();

       return getClient().deleteKey(keyRequest)
           .whenComplete((response, exception) -> {
               if (exception != null) {
                   Throwable cause = exception.getCause();
                   if (cause instanceof ResourceNotFoundException) {
                       throw new CompletionException("The key was not found.",
cause);
                   }
                   throw new CompletionException("Failed to delete key: " +
exception.getMessage(), exception);
               }
               logger.info("The key {} was deleted.", keyName);
           })
           .thenApply(response -> null);
   }


   /**
    * Deletes a map with the specified name.
    *
    * @param mapName the name of the map to be deleted
    * @return a {@link CompletableFuture} that completes when the map deletion is
successful, or throws a {@link CompletionException} if an error occurs
    */
   public CompletableFuture<Void> deleteMap(String mapName) {
       DeleteMapRequest mapRequest = DeleteMapRequest.builder()
           .mapName(mapName)
           .build();
```

```
        return getClient().deleteMap(mapRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof ResourceNotFoundException) {
                        throw new CompletionException("The map was not found.",
cause);
                    }
                    throw new CompletionException("Failed to delete map: " +
exception.getMessage(), exception);
                }
                logger.info("The map {} was deleted.", mapName);
            })
            .thenApply(response -> null);
    }


    /**
     * Deletes a tracker with the specified name.
     *
     * @param trackerName the name of the tracker to be deleted
     * @return a {@link CompletableFuture} that completes when the tracker has been
deleted
     * @throws CompletionException if an error occurs while deleting the tracker
     *                             - if the tracker was not found, a {@link
ResourceNotFoundException} is thrown wrapped in the CompletionException
     *                             - if any other error occurs, a generic
CompletionException is thrown with the error message
     */
    public CompletableFuture<Void> deleteTracker(String trackerName) {
        DeleteTrackerRequest trackerRequest = DeleteTrackerRequest.builder()
            .trackerName(trackerName)
            .build();

        return getClient().deleteTracker(trackerRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof ResourceNotFoundException) {
                        throw new CompletionException("The tracker was not found.",
cause);
                    }
```

```
                    throw new CompletionException("Failed to delete the tracker: " +
 exception.getMessage(), exception);
                }
                logger.info("The tracker {} was deleted.", trackerName);
            })
            .thenApply(response -> null); // Ensures CompletableFuture<Void>
    }


    /**
     * Deletes a route calculator from the system.
     *
     * @param calcName the name of the route calculator to delete
     * @return a {@link CompletableFuture} that completes when the route calculator
 has been deleted
     * @throws CompletionException if an error occurs while deleting the route
 calculator
     *                             - If the route calculator was not found, a {@link
 ResourceNotFoundException} will be thrown
     *                             - If any other error occurs, a generic {@link
 CompletionException} will be thrown
     */
    public CompletableFuture<Void> deleteRouteCalculator(String calcName) {
        DeleteRouteCalculatorRequest calculatorRequest =
 DeleteRouteCalculatorRequest.builder()
            .calculatorName(calcName)
            .build();

        return getClient().deleteRouteCalculator(calculatorRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof ResourceNotFoundException) {
                        throw new CompletionException("The route calculator was not
 found.", cause);
                    }
                    throw new CompletionException("Failed to delete the route
 calculator: " + exception.getMessage(), exception);
                }
                logger.info("The route calculator {} was deleted.", calcName);
            })
            .thenApply(response -> null);
    }
}
```

Azioni

`BatchUpdateDevicePosition`

Il seguente esempio di codice mostra come usare. `BatchUpdateDevicePosition`

SDK per Java 2.x

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
   /**
    * Updates the position of a device in the location tracking system.
    *
    * @param trackerName the name of the tracker associated with the device
    * @param deviceId    the unique identifier of the device
    * @throws RuntimeException if an error occurs while updating the device
position
    */
   public CompletableFuture<BatchUpdateDevicePositionResponse>
updateDevicePosition(String trackerName, String deviceId) {
       double latitude = 37.7749;  // Example: San Francisco
       double longitude = -122.4194;

       DevicePositionUpdate positionUpdate = DevicePositionUpdate.builder()
           .deviceId(deviceId)
           .sampleTime(Instant.now()) // Timestamp of position update.
           .position(Arrays.asList(longitude, latitude)) // AWS requires
[longitude, latitude]
           .build();

       BatchUpdateDevicePositionRequest request =
BatchUpdateDevicePositionRequest.builder()
           .trackerName(trackerName)
           .updates(positionUpdate)
```

```
            .build();

        CompletableFuture<BatchUpdateDevicePositionResponse> futureResponse =
getClient().batchUpdateDevicePosition(request);
        return futureResponse.whenComplete((response, exception) -> {
            if (exception != null) {
                Throwable cause = exception.getCause();
                if (cause instanceof ResourceNotFoundException) {
                    throw new CompletionException("The resource was not found: " +
cause.getMessage(), cause);
                } else {
                    throw new CompletionException("Error updating device position: "
+ exception.getMessage(), exception);
                }
            }
        });
    }
```

- Per i dettagli sull'API, consulta la BatchUpdateDevicePositionsezione AWS SDK for Java 2.x API Reference.

`CalculateRoute`

Il seguente esempio di codice mostra come utilizzare`CalculateRoute`.

SDK per Java 2.x

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
    /**
     * Calculates the distance between two locations asynchronously.
     *
     * @param routeCalcName the name of the route calculator to use
     * @return a {@link CompletableFuture} that will complete with a {@link
CalculateRouteResponse} containing the distance and estimated duration of the route
```

```
     */
    public CompletableFuture<CalculateRouteResponse> calcDistanceAsync(String
routeCalcName) {
        // Define coordinates for Seattle, WA and Vancouver, BC.
        List<Double> departurePosition = Arrays.asList(-122.3321, 47.6062);
        List<Double> arrivePosition = Arrays.asList(-123.1216, 49.2827);

        CalculateRouteRequest request = CalculateRouteRequest.builder()
            .calculatorName(routeCalcName)
            .departurePosition(departurePosition)
            .destinationPosition(arrivePosition)
            .travelMode("Car") // Options: Car, Truck, Walking, Bicycle
            .distanceUnit("Kilometers") // Options: Meters, Kilometers, Miles
            .build();

        return getClient().calculateRoute(request)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof ResourceNotFoundException) {
                        throw new CompletionException("The AWS resource was not
found: " + cause.getMessage(), cause);
                    }
                    throw new CompletionException("Failed to calculate route: " +
exception.getMessage(), exception);
                }
            });
    }
```

- Per i dettagli sull'API, consulta la CalculateRoutesezione AWS SDK for Java 2.x API Reference.

`CreateGeofenceCollection`

Il seguente esempio di codice mostra come utilizzare`CreateGeofenceCollection`.

SDK per Java 2.x

> **Note**
>
> C'è 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 a new geofence collection.
    *
    * @param collectionName the name of the geofence collection to be created
    */
   public CompletableFuture<String> createGeofenceCollection(String collectionName)
{
       CreateGeofenceCollectionRequest collectionRequest =
CreateGeofenceCollectionRequest.builder()
           .collectionName(collectionName)
           .description("Created by using the AWS SDK for Java")
           .build();

       return getClient().createGeofenceCollection(collectionRequest)
           .whenComplete((response, exception) -> {
               if (exception != null) {
                   Throwable cause = exception.getCause();
                   if (cause instanceof ConflictException) {
                       throw new CompletionException("The geofence collection was
not created due to ConflictException.", cause);
                   }
                   throw new CompletionException("Failed to create geofence
collection: " + exception.getMessage(), exception);
               }
           })
           .thenApply(response -> response.collectionArn()); // Return only the ARN
   }
```

- Per i dettagli sull'API, consulta la CreateGeofenceCollectionsezione 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.

```
   /**
    * Creates a new API key with the specified name and restrictions.
    *
    * @param keyName the name of the API key to be created
    * @param mapArn  the Amazon Resource Name (ARN) of the map resource to which
the API key will be associated
    * @return a {@link CompletableFuture} that completes with the Amazon Resource
Name (ARN) of the created API key,
    * or {@code null} if the operation failed
    */
   public CompletableFuture<String> createKey(String keyName, String mapArn) {
       ApiKeyRestrictions keyRestrictions = ApiKeyRestrictions.builder()
           .allowActions("geo:GetMap*")
           .allowResources(mapArn)
           .build();

       CreateKeyRequest request = CreateKeyRequest.builder()
           .keyName(keyName)
           .restrictions(keyRestrictions)
           .noExpiry(true)
           .build();

       return getClient().createKey(request)
           .whenComplete((response, exception) -> {
               if (exception != null) {
                   Throwable cause = exception.getCause();
                   if (cause instanceof AccessDeniedException) {
                       throw new CompletionException("The request was denied
because of insufficient access or permissions.", cause);
                   }
                   throw new CompletionException("Failed to create API key: " +
exception.getMessage(), exception);
               }
```

```
            })
            .thenApply(response -> response.keyArn()); // This will never return
null if the response reaches here
    }
```

- Per i dettagli sull'API, consulta la CreateKeysezione AWS SDK for Java 2.x API Reference.

`CreateMap`

Il seguente esempio di codice mostra come utilizzare`CreateMap`.

SDK per Java 2.x

> **Note**
>
> C'è 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 a new map with the specified name and configuration.
     *
     * @param mapName the name of the map to be created
     * @return a {@link CompletableFuture} that, when completed, will contain the
Amazon Resource Name (ARN) of the created map
     * @throws CompletionException if an error occurs while creating the map, such
as exceeding the service quota
     */
    public CompletableFuture<String> createMap(String mapName) {
        MapConfiguration configuration = MapConfiguration.builder()
            .style("VectorEsriNavigation")
            .build();

        CreateMapRequest mapRequest = CreateMapRequest.builder()
            .mapName(mapName)
            .configuration(configuration)
            .description("A map created using the Java V2 API")
            .build();
```

```
        return getClient().createMap(mapRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof ServiceQuotaExceededException) {
                        throw new CompletionException("The operation was denied
because the request would exceed the maximum quota.", cause);
                    }
                    throw new CompletionException("Failed to create map: " +
exception.getMessage(), exception);
                }
            })
            .thenApply(response -> response.mapArn()); // Return the map ARN
    }
```

- Per i dettagli sull'API, consulta la CreateMapsezione AWS SDK for Java 2.x API Reference.

`CreateRouteCalculator`

Il seguente esempio di codice mostra come utilizzare`CreateRouteCalculator`.

SDK per Java 2.x

> **Note**
>
> C'è 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 a new route calculator with the specified name and data source.
     *
     * @param routeCalcName the name of the route calculator to be created
     */
    public CompletableFuture<CreateRouteCalculatorResponse>
createRouteCalculator(String routeCalcName) {
        String dataSource = "Esri"; // or "Here"
```

```
        CreateRouteCalculatorRequest request =
CreateRouteCalculatorRequest.builder()
            .calculatorName(routeCalcName)
            .dataSource(dataSource)
            .build();

        return getClient().createRouteCalculator(request)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof ConflictException) {
                        throw new CompletionException("A conflict error occurred: "
+ cause.getMessage(), cause);
                    }
                    throw new CompletionException("Failed to create route
calculator: " + exception.getMessage(), exception);
                }
            });
    }
```

- Per i dettagli sull'API, consulta la CreateRouteCalculatorsezione AWS SDK for Java 2.x API Reference.

`CreateTracker`

Il seguente esempio di codice mostra come utilizzare`CreateTracker`.

SDK per Java 2.x

> **Note**
>
> C'è 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 a new tracker resource in your AWS account, which you can use to
track the location of devices.
     *
```

```
     * @param trackerName the name of the tracker to be created
     * @return a {@link CompletableFuture} that, when completed, will contain the
Amazon Resource Name (ARN) of the created tracker
     */
    public CompletableFuture<String> createTracker(String trackerName) {
        CreateTrackerRequest trackerRequest = CreateTrackerRequest.builder()
            .description("Created using the Java V2 SDK")
            .trackerName(trackerName)
            .positionFiltering("TimeBased") // Options: TimeBased, DistanceBased,
AccuracyBased
            .build();

        return getClient().createTracker(trackerRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof ConflictException) {
                        throw new CompletionException("Conflict occurred while
creating tracker: " + cause.getMessage(), cause);
                    }
                    throw new CompletionException("Error creating tracker: " +
exception.getMessage(), exception);
                }
            })
            .thenApply(CreateTrackerResponse::trackerArn); // Return only the
tracker ARN
    }
```

- Per i dettagli sull'API, consulta la CreateTrackersezione AWS SDK for Java 2.x API Reference.

`DeleteGeofenceCollection`

Il seguente esempio di codice mostra come utilizzare`DeleteGeofenceCollection`.

SDK per Java 2.x

> **Note**
>
> C'è 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 geofence collection asynchronously.
    *
    * @param collectionName the name of the geofence collection to be deleted
    * @return a {@link CompletableFuture} that completes when the geofence
collection has been deleted
    */
   public CompletableFuture<Void> deleteGeofenceCollectionAsync(String
collectionName) {
       DeleteGeofenceCollectionRequest collectionRequest =
DeleteGeofenceCollectionRequest.builder()
           .collectionName(collectionName)
           .build();

       return getClient().deleteGeofenceCollection(collectionRequest)
           .whenComplete((response, exception) -> {
               if (exception != null) {
                   Throwable cause = exception.getCause();
                   if (cause instanceof ResourceNotFoundException) {
                       throw new CompletionException("The requested geofence
collection was not found.", cause);
                   }
                   throw new CompletionException("Failed to delete geofence
collection: " + exception.getMessage(), exception);
               }
               logger.info("The geofence collection {} was deleted.",
collectionName);
           })
           .thenApply(response -> null);
   }
```

- Per i dettagli sull'API, consulta la DeleteGeofenceCollectionsezione AWS SDK for Java 2.x API Reference.

`DeleteKey`

Il seguente esempio di codice mostra come utilizzare`DeleteKey`.

SDK per Java 2.x

> **Note**
>
> C'è 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 specified key from the key-value store.
    *
    * @param keyName the name of the key to be deleted
    * @return a {@link CompletableFuture} that completes when the key has been
deleted
    * @throws CompletionException if the key was not found or if an error occurred
during the deletion process
    */
   public CompletableFuture<Void> deleteKey(String keyName) {
       DeleteKeyRequest keyRequest = DeleteKeyRequest.builder()
           .keyName(keyName)
           .build();

       return getClient().deleteKey(keyRequest)
           .whenComplete((response, exception) -> {
               if (exception != null) {
                   Throwable cause = exception.getCause();
                   if (cause instanceof ResourceNotFoundException) {
                       throw new CompletionException("The key was not found.",
cause);
                   }
                   throw new CompletionException("Failed to delete key: " +
exception.getMessage(), exception);
               }
               logger.info("The key {} was deleted.", keyName);
           })
           .thenApply(response -> null);
   }
```

- Per i dettagli sull'API, consulta la DeleteKeysezione AWS SDK for Java 2.x API Reference.

`DeleteMap`

Il seguente esempio di codice mostra come utilizzare`DeleteMap`.

SDK per Java 2.x

> **Note**
>
> C'è 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 map with the specified name.
    *
    * @param mapName the name of the map to be deleted
    * @return a {@link CompletableFuture} that completes when the map deletion is
successful, or throws a {@link CompletionException} if an error occurs
    */
   public CompletableFuture<Void> deleteMap(String mapName) {
       DeleteMapRequest mapRequest = DeleteMapRequest.builder()
           .mapName(mapName)
           .build();

       return getClient().deleteMap(mapRequest)
           .whenComplete((response, exception) -> {
               if (exception != null) {
                   Throwable cause = exception.getCause();
                   if (cause instanceof ResourceNotFoundException) {
                       throw new CompletionException("The map was not found.",
cause);
                   }
                   throw new CompletionException("Failed to delete map: " +
exception.getMessage(), exception);
               }
               logger.info("The map {} was deleted.", mapName);
           })
           .thenApply(response -> null);
   }
```

- Per i dettagli sull'API, consulta la DeleteMapsezione AWS SDK for Java 2.x API Reference.

`DeleteRouteCalculator`

Il seguente esempio di codice mostra come utilizzare`DeleteRouteCalculator`.

SDK per Java 2.x

> **Note**
>
> C'è 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 route calculator from the system.
    *
    * @param calcName the name of the route calculator to delete
    * @return a {@link CompletableFuture} that completes when the route calculator
has been deleted
    * @throws CompletionException if an error occurs while deleting the route
calculator
    *                             - If the route calculator was not found, a {@link
ResourceNotFoundException} will be thrown
    *                             - If any other error occurs, a generic {@link
CompletionException} will be thrown
    */
   public CompletableFuture<Void> deleteRouteCalculator(String calcName) {
       DeleteRouteCalculatorRequest calculatorRequest =
DeleteRouteCalculatorRequest.builder()
           .calculatorName(calcName)
           .build();

       return getClient().deleteRouteCalculator(calculatorRequest)
           .whenComplete((response, exception) -> {
               if (exception != null) {
                   Throwable cause = exception.getCause();
                   if (cause instanceof ResourceNotFoundException) {
                       throw new CompletionException("The route calculator was not
found.", cause);
                   }
```

```
                    throw new CompletionException("Failed to delete the route
calculator: " + exception.getMessage(), exception);
                }
                logger.info("The route calculator {} was deleted.", calcName);
            })
            .thenApply(response -> null);
    }
```

- Per i dettagli sull'API, consulta la DeleteRouteCalculatorsezione AWS SDK for Java 2.x API Reference.

`DeleteTracker`

Il seguente esempio di codice mostra come utilizzare`DeleteTracker`.

SDK per Java 2.x

> **Note**
>
> C'è 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 tracker with the specified name.
     *
     * @param trackerName the name of the tracker to be deleted
     * @return a {@link CompletableFuture} that completes when the tracker has been
deleted
     * @throws CompletionException if an error occurs while deleting the tracker
     *                             - if the tracker was not found, a {@link
ResourceNotFoundException} is thrown wrapped in the CompletionException
     *                             - if any other error occurs, a generic
CompletionException is thrown with the error message
     */
    public CompletableFuture<Void> deleteTracker(String trackerName) {
        DeleteTrackerRequest trackerRequest = DeleteTrackerRequest.builder()
            .trackerName(trackerName)
            .build();
```

```
        return getClient().deleteTracker(trackerRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof ResourceNotFoundException) {
                        throw new CompletionException("The tracker was not found.",
cause);
                    }
                    throw new CompletionException("Failed to delete the tracker: " +
exception.getMessage(), exception);
                }
                logger.info("The tracker {} was deleted.", trackerName);
            })
            .thenApply(response -> null); // Ensures CompletableFuture<Void>
    }
```

- Per i dettagli sull'API, consulta la DeleteTrackersezione AWS SDK for Java 2.x API Reference.

`GetDevicePosition`

Il seguente esempio di codice mostra come utilizzare`GetDevicePosition`.

SDK per Java 2.x

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
    /**
     * Retrieves the position of a device using the provided LocationClient.
     *
     * @param trackerName The name of the tracker associated with the device.
     * @param deviceId    The ID of the device to retrieve the position for.
     * @throws RuntimeException If there is an error fetching the device position.
     */
    public CompletableFuture<GetDevicePositionResponse> getDevicePosition(String
trackerName, String deviceId) {
```

```
        GetDevicePositionRequest request = GetDevicePositionRequest.builder()
            .trackerName(trackerName)
            .deviceId(deviceId)
            .build();

        return getClient().getDevicePosition(request)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof ResourceNotFoundException) {
                        throw new CompletionException("The AWS resource was not
found: " + cause.getMessage(), cause);
                    }
                    throw new CompletionException("Error fetching device position: "
+ exception.getMessage(), exception);
                }
            });
    }
```

- Per i dettagli sull'API, consulta la GetDevicePositionsezione AWS SDK for Java 2.x API Reference.

`PutGeofence`

Il seguente esempio di codice mostra come utilizzare`PutGeofence`.

SDK per Java 2.x

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
    /**
     * Adds a new geofence to the specified collection.
     *
     * @param collectionName the name of the geofence collection to add the geofence
to
```

```
     * @param geoId          the unique identifier for the geofence
     */
    public CompletableFuture<PutGeofenceResponse> putGeofence(String collectionName,
String geoId) {
        // Define the geofence geometry (polygon).
        GeofenceGeometry geofenceGeometry = GeofenceGeometry.builder()
            .polygon(List.of(
                List.of(
                    List.of(-122.3381, 47.6101), // First point
                    List.of(-122.3281, 47.6101),
                    List.of(-122.3281, 47.6201),
                    List.of(-122.3381, 47.6201),
                    List.of(-122.3381, 47.6101) // Closing the polygon
                )
            ))
            .build();

        PutGeofenceRequest geofenceRequest = PutGeofenceRequest.builder()
            .collectionName(collectionName) // Specify the collection.
            .geofenceId(geoId) // Unique ID for the geofence.
            .geometry(geofenceGeometry)
            .build();

        return getClient().putGeofence(geofenceRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof ValidationException) {
                        throw new CompletionException("Validation error while
creating geofence: " + cause.getMessage(), cause);
                    }
                    throw new CompletionException("Error creating geofence: " +
exception.getMessage(), exception);
                }
            });
    }
```

- Per i dettagli sull'API, consulta la PutGeofencesezione AWS SDK for Java 2.x API Reference.

Location Service Places: esempi con SDK for Java 2.x

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando Location Service Places. 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.

Ogni esempio include un collegamento al codice sorgente completo, in cui è possibile trovare istruzioni su come configurare ed eseguire il codice nel contesto.

Argomenti

- Azioni

Azioni

`ReverseGeocode`

Il seguente esempio di codice mostra come utilizzare`ReverseGeocode`.

SDK per Java 2.x

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
   /**
    * Performs reverse geocoding using the AWS Geo Places API.
    * Reverse geocoding is the process of converting geographic coordinates
(latitude and longitude) to a human-readable address.
    * This method uses the latitude and longitude of San Francisco as the input,
and prints the resulting address.
    */
   public CompletableFuture<ReverseGeocodeResponse> reverseGeocode() {
       double latitude = 37.7749;  // San Francisco
```

```
        double longitude = -122.4194;
        logger.info("Use latitude 37.7749 and longitude -122.4194");

        // AWS expects [longitude, latitude].
        List<Double> queryPosition = List.of(longitude, latitude);
        ReverseGeocodeRequest request = ReverseGeocodeRequest.builder()
            .queryPosition(queryPosition)
            .build();
        CompletableFuture<ReverseGeocodeResponse> futureResponse =
            getGeoPlacesClient().reverseGeocode(request);

        return futureResponse.whenComplete((response, exception) -> {
            if (exception != null) {
                Throwable cause = exception.getCause();
                if (cause instanceof
software.amazon.awssdk.services.geoplaces.model.ValidationException) {
                    throw new CompletionException("A validation error occurred: " +
cause.getMessage(), cause);
                }
                throw new CompletionException("Error performing reverse geocoding",
exception);
            }

            response.resultItems().forEach(result ->
                logger.info("The address is: " + result.address().label())
            );
        });
    }
```

- Per i dettagli sull'API, consulta la ReverseGeocodesezione AWS SDK for Java 2.x API Reference.

`SearchNearby`

Il seguente esempio di codice mostra come utilizzare`SearchNearby`.

SDK per Java 2.x

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
    /**
     * Performs a nearby places search based on the provided geographic coordinates
 (latitude and longitude).
     * The method sends an asynchronous request to search for places within a 1-
kilometer radius of the specified location.
     * The results are processed and printed once the search completes successfully.
     */
    public CompletableFuture<SearchNearbyResponse> searchNearBy() {
        double latitude = 37.7749;  // San Francisco
        double longitude = -122.4194;
        List<Double> queryPosition = List.of(longitude, latitude);

        // Set up the request for searching nearby places.
        SearchNearbyRequest request = SearchNearbyRequest.builder()
            .queryPosition(queryPosition)  // Set the position
            .queryRadius(1000L)  // Radius in meters (1000 meters = 1 km).
            .build();

        return getGeoPlacesClient().searchNearby(request)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof
 software.amazon.awssdk.services.geoplaces.model.ValidationException) {
                        throw new CompletionException("A validation error occurred:
 " + cause.getMessage(), cause);
                    }
                    throw new CompletionException("Error performing place search",
 exception);
                }

                // Process the response and print the results.
                response.resultItems().forEach(result -> {
```

```
                    logger.info("Place Name: " + result.placeType().name());
                    logger.info("Address: " + result.address().label());
                    logger.info("Distance: " + result.distance() + " meters");
                    logger.info("-------------------------");
                });
            });
    }
```

- Per i dettagli sull'API, consulta la SearchNearbysezione AWS SDK for Java 2.x API Reference.

`SearchText`

Il seguente esempio di codice mostra come utilizzare`SearchText`.

SDK per Java 2.x

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
    /**
     * Searches for a place using the provided search query and prints the detailed
information of the first result.
     *
     * @param searchQuery the search query to be used for the place search (ex,
coffee shop)
     */
    public CompletableFuture<Void> searchText(String searchQuery) {
        double latitude = 37.7749;  // San Francisco
        double longitude = -122.4194;
        List<Double> queryPosition = List.of(longitude, latitude);

        SearchTextRequest request = SearchTextRequest.builder()
                .queryText(searchQuery)
                .biasPosition(queryPosition)
                .build();

        return getGeoPlacesClient().searchText(request)
```

```
                .thenCompose(response -> {
                    if (response.resultItems().isEmpty()) {
                        logger.info("No places found.");
                        return CompletableFuture.completedFuture(null);
                    }

                    // Get the first place ID
                    String placeId = response.resultItems().get(0).placeId();
                    logger.info("Found Place with id: " + placeId);

                    // Fetch detailed info using getPlace
                    GetPlaceRequest getPlaceRequest = GetPlaceRequest.builder()
                            .placeId(placeId)
                            .build();

                    return getGeoPlacesClient().getPlace(getPlaceRequest)
                            .thenAccept(placeResponse -> {
                                logger.info("Detailed Place Information:");
                                logger.info("Name: " +
 placeResponse.placeType().name());
                                logger.info("Address: " +
 placeResponse.address().label());

                                if (placeResponse.foodTypes() != null && !
placeResponse.foodTypes().isEmpty()) {
                                    logger.info("Food Types:");
                                    placeResponse.foodTypes().forEach(foodType -> {
                                        logger.info("  - " + foodType);
                                    });
                                } else {
                                    logger.info("No food types available.");
                                }
                                logger.info("-------------------------");
                            });
                })
                .exceptionally(exception -> {
                    Throwable cause = exception.getCause();
                    if (cause instanceof
 software.amazon.awssdk.services.geoplaces.model.ValidationException) {
                        throw new CompletionException("A validation error occurred:
 " + cause.getMessage(), cause);
                    }
                    throw new CompletionException("Error performing place search",
 exception);
```

```
                });
    }
```

- Per i dettagli sull'API, consulta la SearchTextsezione AWS SDK for Java 2.x API Reference.

Marketplace AWS Esempi di API di catalogo che utilizzano 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 Marketplace AWS Catalog.

Ogni esempio include un collegamento al codice sorgente completo, in cui è possibile trovare istruzioni su come configurare ed eseguire il codice nel contesto.

Argomenti

- Prodotti AMI
- Offerte per i partner di canale
- Prodotti container
- Entità
- Offers (Offerte)
- Prodotti
- Autorizzazione alla rivendita
- Prodotti SaaS
- Utilità

Prodotti AMI

Aggiungi una dimensione a un prodotto AMI esistente e aggiorna i termini e i prezzi dell'offerta

Il seguente esempio di codice mostra come aggiungere una dimensione a un prodotto AMI esistente e aggiornare i termini di prezzo dell'offerta.

SDK per Java 2.x

> **Note**
>
> C'è di più su GitHub. Trova l'esempio completo e scopri come configurarlo ed eseguirlo nel repository Marketplace AWS API Reference Code Library.

Per eseguire questo esempio, passa il seguente changeset JSON a **`RunChangesets`** in Utilities per avviare un changeset dalla sezione Utilities.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "AddDimensions",
            "Entity": {
                "Identifier": "prod-1111111111111",
                "Type": "AmiProduct@1.0"
            },
            "DetailsDocument": [
                {
                    "Key": "m7g.8xlarge",
                    "Description": "m7g.8xlarge",
                    "Name": "m7g.8xlarge",
                    "Types": [
                        "Metered"
                    ],
                    "Unit": "Hrs"
                }
            ]
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "offer-1111111111111"
            },
            "DetailsDocument": {
                "PricingModel": "Usage",
                "Terms": [
                    {
```

```
                        "Type": "UsageBasedPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "RateCard": [
                                    {
                                        "DimensionKey": "m5.large",
                                        "Price": "0.15"
                                    },
                                    {
                                        "DimensionKey": "m7g.4xlarge",
                                        "Price": "0.45"
                                    },
                                    {
                                        "DimensionKey": "m7g.2xlarge",
                                        "Price": "0.45"
                                    },
                                    {
                                        "DimensionKey": "m7g.8xlarge",
                                        "Price": "0.55"
                                    }
                                ]
                            }
                        ]
                    }
                ]
            }
        }
    ]
}
```

- Per i dettagli sull'API, consulta API Reference. StartChangeSetAWS SDK for Java 2.x

Aggiungere una regione in cui viene distribuito un prodotto AMI

Il seguente esempio di codice mostra come aggiungere una regione in cui viene distribuito un prodotto AMI.

SDK per Java 2.x

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri come configurarlo ed eseguirlo nel repository Marketplace AWS API Reference Code Library.

Per eseguire questo esempio, passa il seguente changeset JSON a **`RunChangesets`** in Utilities per avviare un changeset dalla sezione Utilities.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "AddRegions",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "prod-1111111111111"
            },
            "DetailsDocument": {
                "Regions": [
                    "us-east-2",
                    "us-west-2"
                ]
            }
        }
    ]
}
```

- Per i dettagli sull'API, consulta API Reference. StartChangeSetAWS SDK for Java 2.x

Crea un prodotto AMI pubblico o limitato e un'offerta pubblica con prezzi orari annuali

Il seguente esempio di codice mostra come creare un prodotto AMI pubblico o limitato e un'offerta pubblica con prezzi orari annuali. Questo esempio crea un EULA standard o personalizzato.

SDK per Java 2.x

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri come configurarlo ed eseguirlo nel repository Marketplace AWS API Reference Code Library.

Per eseguire questo esempio, passa il seguente changeset JSON a **`RunChangesets`** in Utilities per avviare un changeset dalla sezione Utilities.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateProduct",
            "ChangeName": "CreateProductChange",
            "Entity": {
                "Type": "AmiProduct@1.0"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "ProductTitle": "Sample product",
                "ShortDescription": "Brief description",
                "LongDescription": "Detailed description",
                "Highlights": [
                    "Sample highlight"
                ],
                "SearchKeywords": [
                    "Sample keyword"
                ],
                "Categories": [
                    "Operating Systems"
                ],
                "LogoUrl": "https://s3.amazonaws.com/logos/sample.png",
```

```
                "VideoUrls": [
                    "https://sample.amazonaws.com/awsmp-video-1"
                ],
                "AdditionalResources": []
            }
        },
        {
            "ChangeType": "AddRegions",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Regions": [
                    "us-east-1"
                ]
            }
        },
        {
            "ChangeType": "AddInstanceTypes",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "InstanceTypes": [
                    "t2.micro"
                ]
            }
        },
        {
            "ChangeType": "AddDeliveryOptions",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Version": {
                    "VersionTitle": "Test AMI Version1.0",
                    "ReleaseNotes": "Test AMI Version"
                },
                "DeliveryOptions": [
                    {
                        "Details": {
```

```
                            "AmiDeliveryOptionDetails": {
                                "AmiSource": {
                                    "AmiId": "ami-1111111111111111",
                                    "AccessRoleArn":
"arn:aws:iam::111111111111:role/AWSMarketplaceAmiIngestion",
                                    "UserName": "ec2-user",
                                    "OperatingSystemName": "AMAZONLINUX",
                                    "OperatingSystemVersion": "10.0.14393",
                                    "ScanningPort": 22
                                },
                                "UsageInstructions": "Test AMI Version",
                                "RecommendedInstanceType": "t2.micro",
                                "SecurityGroups": [
                                    {
                                        "IpProtocol": "tcp",
                                        "IpRanges": [
                                            "0.0.0.0/0"
                                        ],
                                        "FromPort": 10,
                                        "ToPort": 22
                                    }
                                ]
                            }
                        }
                    }
                ]
            }
        },
        {
            "ChangeType": "AddDimensions",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": [
                {
                    "Key": "t2.micro",
                    "Description": "t2.micro",
                    "Name": "t2.micro",
                    "Types": [
                        "Metered"
                    ],
                    "Unit": "Hrs"
                }
```

```
            ]
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111",
                        "222222222222"
                    ]
                }
            }
        },
        {
            "ChangeType": "ReleaseProduct",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "CreateOffer",
            "ChangeName": "CreateOfferChange",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "DetailsDocument": {
                "ProductId": "$CreateProductChange.Entity.Identifier"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test public offer for AmiProduct using AWS Marketplace API
Reference Code",
```

```
                "Description": "Test public offer with hourly-annual pricing for
AmiProduct using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Usage",
                "Terms": [
                    {
                        "Type": "UsageBasedPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.micro",
                                        "Price": "0.15"
                                    }
                                ]
                            }
                        ]
                    },
                    {
                        "Type": "ConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P365D"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.micro",
                                        "Price": "150"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
```

```
                                "QuantityConfiguration": "Allowed"
                            }
                        }
                    ]
                }
            ]
        }
    },
    {
        "ChangeType": "UpdateLegalTerms",
        "Entity": {
            "Type": "Offer@1.0",
            "Identifier": "$CreateOfferChange.Entity.Identifier"
        },
        "DetailsDocument": {
            "Terms": [
                {
                    "Type": "LegalTerm",
                    "Documents": [
                        {
                            "Type": "StandardEula",
                            "Version": "2022-07-14"
                        }
                    ]
                }
            ]
        }
    },
    {
        "ChangeType": "UpdateSupportTerms",
        "Entity": {
            "Type": "Offer@1.0",
            "Identifier": "$CreateOfferChange.Entity.Identifier"
        },
        "DetailsDocument": {
            "Terms": [
                {
                    "Type": "SupportTerm",
                    "RefundPolicy": "Absolutely no refund, period."
                }
            ]
        }
    },
    {
```

```
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

- Per i dettagli sull'API, consulta API Reference. StartChangeSetAWS SDK for Java 2.x

Crea un prodotto AMI pubblico o limitato e un'offerta pubblica con prezzi mensili orari

Il seguente esempio di codice mostra come creare un prodotto AMI pubblico o limitato e un'offerta pubblica con prezzi orari mensili. Questo esempio crea un EULA standard o personalizzato.

SDK per Java 2.x

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri come configurarlo ed eseguirlo nel repository Marketplace AWS API Reference Code Library.

Per eseguire questo esempio, passa il seguente changeset JSON a **`RunChangesets`** in Utilities per avviare un changeset dalla sezione Utilities.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateProduct",
            "ChangeName": "CreateProductChange",
            "Entity": {
                "Type": "AmiProduct@1.0"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
```

```
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "ProductTitle": "Sample product",
                "ShortDescription": "Brief description",
                "LongDescription": "Detailed description",
                "Highlights": [
                    "Sample highlight"
                ],
                "SearchKeywords": [
                    "Sample keyword"
                ],
                "Categories": [
                    "Operating Systems"
                ],
                "LogoUrl": "https://s3.amazonaws.com/logos/sample.png",
                "VideoUrls": [
                    "https://sample.amazonaws.com/awsmp-video-1"
                ],
                "AdditionalResources": []
            }
        },
        {
            "ChangeType": "AddRegions",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Regions": [
                    "us-east-1"
                ]
            }
        },
        {
            "ChangeType": "AddInstanceTypes",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "InstanceTypes": [
                    "t2.micro"
```

```
                ]
            }
        },
        {
            "ChangeType": "AddDeliveryOptions",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Version": {
                    "VersionTitle": "Test AMI Version1.0",
                    "ReleaseNotes": "Test AMI Version"
                },
                "DeliveryOptions": [
                    {
                        "Details": {
                            "AmiDeliveryOptionDetails": {
                                "AmiSource": {
                                    "AmiId": "ami-11111111111111111",
                                    "AccessRoleArn":
"arn:aws:iam::111111111111:role/AWSMarketplaceAmiIngestion",
                                    "UserName": "ec2-user",
                                    "OperatingSystemName": "AMAZONLINUX",
                                    "OperatingSystemVersion": "10.0.14393",
                                    "ScanningPort": 22
                                },
                                "UsageInstructions": "Test AMI Version",
                                "RecommendedInstanceType": "t2.micro",
                                "SecurityGroups": [
                                    {
                                        "IpProtocol": "tcp",
                                        "IpRanges": [
                                            "0.0.0.0/0"
                                        ],
                                        "FromPort": 10,
                                        "ToPort": 22
                                    }
                                ]
                            }
                        }
                    }
                ]
            }
```

```
        },
        {
            "ChangeType": "AddDimensions",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": [
                {
                    "Key": "t2.micro",
                    "Description": "t2.micro",
                    "Name": "t2.micro",
                    "Types": [
                        "Metered"
                    ],
                    "Unit": "Hrs"
                }
            ]
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111",
                        "222222222222"
                    ]
                }
            }
        },
        {
            "ChangeType": "ReleaseProduct",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "CreateOffer",
```

```
            "ChangeName": "CreateOfferChange",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "DetailsDocument": {
                "ProductId": "$CreateProductChange.Entity.Identifier"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test public offer for AmiProduct using AWS Marketplace API
Reference Code",
                "Description": "Test public offer with hourly-monthly pricing for
AmiProduct using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Usage",
                "Terms": [
                    {
                        "Type": "UsageBasedPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.micro",
                                        "Price": "0.15"
                                    }
                                ]
                            }
                        ]
                    },
```

```
                {
                    "Type": "RecurringPaymentTerm",
                    "CurrencyCode": "USD",
                    "BillingPeriod": "Monthly",
                    "Price": "15.0"
                }
            ]
        }
    },
    {
        "ChangeType": "UpdateLegalTerms",
        "Entity": {
            "Type": "Offer@1.0",
            "Identifier": "$CreateOfferChange.Entity.Identifier"
        },
        "DetailsDocument": {
            "Terms": [
                {
                    "Type": "LegalTerm",
                    "Documents": [
                        {
                            "Type": "StandardEula",
                            "Version": "2022-07-14"
                        }
                    ]
                }
            ]
        }
    },
    {
        "ChangeType": "UpdateSupportTerms",
        "Entity": {
            "Type": "Offer@1.0",
            "Identifier": "$CreateOfferChange.Entity.Identifier"
        },
        "DetailsDocument": {
            "Terms": [
                {
                    "Type": "SupportTerm",
                    "RefundPolicy": "Absolutely no refund, period."
                }
            ]
        }
    },
```

```
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

- Per i dettagli sull'API, consulta API Reference. StartChangeSetAWS SDK for Java 2.x

Crea un prodotto AMI pubblico o limitato e un'offerta pubblica con prezzi orari

Il seguente esempio di codice mostra come creare un prodotto AMI pubblico o limitato e un'offerta pubblica con prezzi orari. Questo esempio crea un EULA standard o personalizzato.

SDK per Java 2.x

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri come configurarlo ed eseguirlo nel repository Marketplace AWS API Reference Code Library.

Per eseguire questo esempio, passa il seguente changeset JSON a **`RunChangesets`** in Utilities per avviare un changeset dalla sezione Utilities.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateProduct",
            "ChangeName": "CreateProductChange",
            "Entity": {
                "Type": "AmiProduct@1.0"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdateInformation",
```

```
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "ProductTitle": "Sample product",
                "ShortDescription": "Brief description",
                "LongDescription": "Detailed description",
                "Highlights": [
                    "Sample highlight"
                ],
                "SearchKeywords": [
                    "Sample keyword"
                ],
                "Categories": [
                    "Operating Systems"
                ],
                "LogoUrl": "https://s3.amazonaws.com/logos/sample.png",
                "VideoUrls": [
                    "https://sample.amazonaws.com/awsmp-video-1"
                ],
                "AdditionalResources": []
            }
        },
        {
            "ChangeType": "AddRegions",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Regions": [
                    "us-east-1"
                ]
            }
        },
        {
            "ChangeType": "AddInstanceTypes",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "InstanceTypes": [
```

```
                    "t2.micro"
                ]
            }
        },
        {
            "ChangeType": "AddDeliveryOptions",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Version": {
                    "VersionTitle": "Test AMI Version1.0",
                    "ReleaseNotes": "Test AMI Version"
                },
                "DeliveryOptions": [
                    {
                        "Details": {
                            "AmiDeliveryOptionDetails": {
                                "AmiSource": {
                                    "AmiId": "ami-1111111111111111",
                                    "AccessRoleArn":
"arn:aws:iam::111111111111:role/AWSMarketplaceAmiIngestion",
                                    "UserName": "ec2-user",
                                    "OperatingSystemName": "AMAZONLINUX",
                                    "OperatingSystemVersion": "10.0.14393",
                                    "ScanningPort": 22
                                },
                                "UsageInstructions": "Test AMI Version",
                                "RecommendedInstanceType": "t2.micro",
                                "SecurityGroups": [
                                    {
                                        "IpProtocol": "tcp",
                                        "IpRanges": [
                                            "0.0.0.0/0"
                                        ],
                                        "FromPort": 10,
                                        "ToPort": 22
                                    }
                                ]
                            }
                        }
                    }
                ]
```

```
            }
        },
        {
            "ChangeType": "AddDimensions",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": [
                {
                    "Key": "t2.micro",
                    "Description": "t2.micro",
                    "Name": "t2.micro",
                    "Types": [
                        "Metered"
                    ],
                    "Unit": "Hrs"
                }
            ]
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111",
                        "222222222222"
                    ]
                }
            }
        },
        {
            "ChangeType": "ReleaseProduct",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
```

```
            "ChangeType": "CreateOffer",
            "ChangeName": "CreateOfferChange",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "DetailsDocument": {
                "ProductId": "$CreateProductChange.Entity.Identifier"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test public offer for AmiProduct using AWS Marketplace API
Reference Code",
                "Description": "Test public offer with hourly pricing for AmiProduct
using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Usage",
                "Terms": [
                    {
                        "Type": "UsageBasedPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.micro",
                                        "Price": "0.15"
                                    }
                                ]
                            }
                        ]
```

```
                }
            ]
        }
    },
    {
        "ChangeType": "UpdateLegalTerms",
        "Entity": {
            "Type": "Offer@1.0",
            "Identifier": "$CreateOfferChange.Entity.Identifier"
        },
        "DetailsDocument": {
            "Terms": [
                {
                    "Type": "LegalTerm",
                    "Documents": [
                        {
                            "Type": "StandardEula",
                            "Version": "2022-07-14"
                        }
                    ]
                }
            ]
        }
    },
    {
        "ChangeType": "UpdateSupportTerms",
        "Entity": {
            "Type": "Offer@1.0",
            "Identifier": "$CreateOfferChange.Entity.Identifier"
        },
        "DetailsDocument": {
            "Terms": [
                {
                    "Type": "SupportTerm",
                    "RefundPolicy": "Absolutely no refund, period."
                }
            ]
        }
    },
    {
        "ChangeType": "ReleaseOffer",
        "Entity": {
            "Type": "Offer@1.0",
            "Identifier": "$CreateOfferChange.Entity.Identifier"
```

```
            },
            "DetailsDocument": {}
        }
    ]
}
```

- Per i dettagli sull'API, consulta API Reference. StartChangeSetAWS SDK for Java 2.x

Crea una bozza di prodotto AMI con una bozza di offerta pubblica

Il seguente esempio di codice mostra come creare una bozza di prodotto AMI con una bozza di offerta pubblica.

SDK per Java 2.x

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri come configurarlo ed eseguirlo nel repository Marketplace AWS API Reference Code Library.

Per eseguire questo esempio, passa il seguente changeset JSON a **`RunChangesets`** in Utilities per avviare un changeset dalla sezione Utilities.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateProduct",
            "ChangeName": "CreateProductChange",
            "Entity": {
                "Type": "AmiProduct@1.0"
            },
            "DetailsDocument": {
                "ProductTitle": "Sample product"
            }
        },
        {
            "ChangeType": "CreateOffer",
            "ChangeName": "CreateOfferChange",
            "Entity": {
```

```
                "Type": "Offer@1.0"
            },
            "DetailsDocument": {
                "ProductId": "$CreateProductChange.Entity.Identifier",
                "Name": "Test Offer"
            }
        }
    ]
}
```

- Per i dettagli sull'API, consulta API Reference. StartChangeSetAWS SDK for Java 2.x

Limita una regione in cui viene distribuito un prodotto AMI

Il seguente esempio di codice mostra come limitare una regione in cui viene distribuito un prodotto AMI.

SDK per Java 2.x

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri come configurarlo ed eseguirlo nel repository Marketplace AWS API Reference Code Library.

Per eseguire questo esempio, passa il seguente changeset JSON a **`RunChangesets`** in Utilities per avviare un changeset dalla sezione Utilities.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "RestrictRegions",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "prod-1111111111111"
            },
            "DetailsDocument": {
                "Regions": [
                    "us-west-2"
                ]
```

```
            }
        }
    ]
}
```

- Per i dettagli sull'API, consulta API Reference. StartChangeSetAWS SDK for Java 2.x

Limita la visibilità del prodotto

Il seguente esempio di codice mostra come limitare la visibilità del prodotto.

SDK per Java 2.x

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri come configurarlo ed eseguirlo nel repository Marketplace AWS API Reference Code Library.

Per eseguire questo esempio, passa il seguente changeset JSON a **`RunChangesets`** in Utilities per avviare un changeset dalla sezione Utilities.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "UpdateVisibility",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "prod-1111111111111"
            },
            "DetailsDocument": {
                "TargetVisibility": "Restricted"
            }
        }
    ]
}
```

- Per i dettagli sull'API, consulta API Reference. StartChangeSetAWS SDK for Java 2.x

Specificare se gli asset AMI vengono distribuiti in nuove regioni

Il seguente esempio di codice mostra come specificare se gli asset AMI vengono distribuiti in nuove regioni create da AWS per supportare le regioni future.

SDK per Java 2.x

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri come configurarlo ed eseguirlo nel repository Marketplace AWS API Reference Code Library.

Per eseguire questo esempio, passa il seguente changeset JSON a **`RunChangesets`** in Utilities per avviare un changeset dalla sezione Utilities.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "UpdateFutureRegionSupport",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "prod-1111111111111"
            },
            "DetailsDocument": {
                "FutureRegionSupport": {
                    "SupportedRegions": [
                        "All"
                    ]
                }
            }
        }
    ]
}
```

- Per i dettagli sull'API, consulta API Reference. StartChangeSetAWS SDK for Java 2.x

Offerte per i partner di canale

Crea una bozza di CPPO per qualsiasi tipo di prodotto

Il seguente esempio di codice mostra come creare una bozza di CPPO per qualsiasi tipo di prodotto in modo da poterla esaminare internamente prima di pubblicarla agli acquirenti.

SDK per Java 2.x

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri come configurarlo ed eseguirlo nel repository Marketplace AWS API Reference Code Library.

Per eseguire questo esempio, passa il seguente changeset JSON a **`RunChangesets`** in Utilities per avviare un changeset dalla sezione Utilities.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateOfferUsingResaleAuthorization",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "DetailsDocument": {
                "ResaleAuthorizationId": "11111111-1111-1111-1111-111111111111",
                "Name": "Test Offer",
                "Description": "Test product"
            }
        }
    ]
}
```

- Per i dettagli sull'API, consulta API Reference. StartChangeSetAWS SDK for Java 2.x

Crea un'autorizzazione alla rivendita, un'offerta privata sostitutiva con prezzi contrattuali

Il seguente esempio di codice mostra come creare un'offerta privata sostitutiva con autorizzazione alla rivendita a partire da un accordo esistente con prezzi contrattuali.

SDK per Java 2.x

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri come configurarlo ed eseguirlo nel repository Marketplace AWS API Reference Code Library.

Per eseguire questo esempio, passa il seguente changeset JSON a **`RunChangesets`** in Utilities per avviare un changeset dalla sezione Utilities.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType" : "CreateReplacementOfferUsingResaleAuthorization",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "ChangeName": "CreateReplacementOfferResaleAuth",
            "DetailsDocument": {
                "AgreementId": "agmt-11111111111111111111111",
                "ResaleAuthorizationId": "resaleauthz-1111111111111"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateReplacementOfferResaleAuth.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test replacement offer for SaaSProduct using AWS
 Marketplace API Reference Codes",
                "Description": "Test private resale replacement offer with contract
 pricing for SaaSProduct"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
```

```
                "Identifier": "$CreateReplacementOfferResaleAuth.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "FixedUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "Price": "0.0",
                        "Duration": "P12M",
                        "Grants": [
                            {
                                "DimensionKey": "BasicService",
                                "MaxQuantity": 2
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateValidityTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateReplacementOfferResaleAuth.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "ValidityTerm",
                        "AgreementEndDate": "2024-01-30"
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdatePaymentScheduleTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateReplacementOfferResaleAuth.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
```

```
                    "Type": "PaymentScheduleTerm",
                    "CurrencyCode": "USD",
                    "Schedule": [
                        {
                            "ChargeDate": "2024-01-01",
                            "ChargeAmount": "0"
                        }
                    ]
                }
            ]
        }
    },
    {
        "ChangeType": "UpdateLegalTerms",
        "Entity": {
            "Type": "Offer@1.0",
            "Identifier": "$CreateReplacementOfferResaleAuth.Entity.Identifier"
        },
        "DetailsDocument": {
            "Terms": [
                {
                    "Type": "LegalTerm",
                    "Documents": [
                        {
                            "Type": "StandardEula",
                            "Version": "2022-07-14"
                        }
                    ]
                }
            ]
        }
    },
    {
        "ChangeType": "UpdateAvailability",
        "Entity": {
            "Type": "Offer@1.0",
            "Identifier": "$CreateReplacementOfferResaleAuth.Entity.Identifier"
        },
        "DetailsDocument": {
            "AvailabilityEndDate": "2023-12-31"
        }
    },
    {
        "ChangeType": "ReleaseOffer",
```

```
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateReplacementOfferResaleAuth.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

- Per i dettagli sull'API, consulta API Reference. StartChangeSetAWS SDK for Java 2.x

Elenca tutti i prodotti CPPOs creati da un partner di canale

Il seguente esempio di codice mostra come elencare tutto ciò che CPPOs è stato creato da un partner di canale.

SDK per Java 2.x

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri come configurarlo ed eseguirlo nel repository Marketplace AWS API Reference Code Library.

```
package com.example.awsmarketplace.catalogapi;

import java.util.ArrayList;
import java.util.List;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.*;
import com.example.awsmarketplace.utils.ReferenceCodesUtils;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.core.document.Document;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.services.marketplacecatalog.MarketplaceCatalogClient;
import
 software.amazon.awssdk.services.marketplacecatalog.model.DescribeEntityRequest;
import
 software.amazon.awssdk.services.marketplacecatalog.model.DescribeEntityResponse;
import software.amazon.awssdk.services.marketplacecatalog.model.EntitySummary;
import software.amazon.awssdk.services.marketplacecatalog.model.ListEntitiesRequest;
```

```
import
 software.amazon.awssdk.services.marketplacecatalog.model.ListEntitiesResponse;

public class ListAllCppoOffers {

 /*
  * List all CPPOs created by a channel partner
  */
 public static void main(String[] args) {

  List<String> cppoOfferIds = getAllCppoOfferIds();

  ReferenceCodesUtils.formatOutput(cppoOfferIds);
 }

 public static List<String> getAllCppoOfferIds() {
  MarketplaceCatalogClient marketplaceCatalogClient =
    MarketplaceCatalogClient.builder()
    .httpClient(ApacheHttpClient.builder().build())
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

  // get all offer entity ids
  List<String> entityIdList = new ArrayList<String>();

  ListEntitiesRequest listEntitiesRequest =
    ListEntitiesRequest.builder()
    .catalog(AWS_MP_CATALOG)
    .entityType(ENTITY_TYPE_OFFER)
    .maxResults(10)
    .nextToken(null)
    .build();

  ListEntitiesResponse listEntitiesResponse =
 marketplaceCatalogClient.listEntities(listEntitiesRequest);

  for (EntitySummary entitySummary : listEntitiesResponse.entitySummaryList()) {
   entityIdList.add(entitySummary.entityId());
  }

  while (listEntitiesResponse.nextToken() != null) {
   listEntitiesRequest =
     ListEntitiesRequest.builder()
     .catalog(AWS_MP_CATALOG)
```

```
      .entityType(ENTITY_TYPE_OFFER)
      .maxResults(10)
      .nextToken(listEntitiesResponse.nextToken())
      .build();
    listEntitiesResponse =
  marketplaceCatalogClient.listEntities(listEntitiesRequest);

    for (EntitySummary entitySummary : listEntitiesResponse.entitySummaryList()) {
     entityIdList.add(entitySummary.entityId());
    }
   }

   // filter for CPPO offers: ResaleAuthorizationId exists in Details

   List<String> cppoOfferIds = new ArrayList<String>();

   for (String entityId : entityIdList) {
    DescribeEntityRequest describeEntityRequest =
      DescribeEntityRequest.builder()
      .catalog(AWS_MP_CATALOG)
      .entityId(entityId)
      .build();
    DescribeEntityResponse describeEntityResponse =
  marketplaceCatalogClient.describeEntity(describeEntityRequest);

    Document resaleAuthorizationDocument =
  describeEntityResponse.detailsDocument().asMap().get(ATTRIBUTE_RESALE_AUTHORIZATION_ID);
    String resaleAuthorizationId = resaleAuthorizationDocument != null ?
  resaleAuthorizationDocument.asString() : "";

    if (!resaleAuthorizationId.isEmpty()) {
        cppoOfferIds.add(resaleAuthorizationId);
    }
   }
   return cppoOfferIds;
  }

}
```

- Per i dettagli sull'API, consulta la sezione ListEntities AWS SDK for Java 2.xAPI Reference.

Elenca tutte le autorizzazioni di rivendita condivise disponibili per un partner di canale

Il seguente esempio di codice mostra come elencare tutte le autorizzazioni di rivendita condivise disponibili per un partner di canale.

SDK per Java 2.x

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri come configurarlo ed eseguirlo nel repository Marketplace AWS API Reference Code Library.

```
package com.example.awsmarketplace.catalogapi;

import java.util.ArrayList;
import java.util.List;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.*;
import com.example.awsmarketplace.utils.ReferenceCodesUtils;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.services.marketplacecatalog.MarketplaceCatalogClient;
import software.amazon.awssdk.services.marketplacecatalog.model.ListEntitiesRequest;
import
 software.amazon.awssdk.services.marketplacecatalog.model.ListEntitiesResponse;

public class ListAllSharedResaleAuthorizations {

 /*
  * list all resale authorizations shared to an account
  */
 public static void main(String[] args) {

  List<ListEntitiesResponse> responseList = getListEntityResponseList();
  ReferenceCodesUtils.formatOutput(responseList);
 }

 public static List<ListEntitiesResponse> getListEntityResponseList() {
  MarketplaceCatalogClient marketplaceCatalogClient =
    MarketplaceCatalogClient.builder()
```

```
    .httpClient(ApacheHttpClient.builder().build())
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

  List<ListEntitiesResponse> responseList = new ArrayList<ListEntitiesResponse>();

  ListEntitiesRequest listEntitiesRequest =
    ListEntitiesRequest.builder()
    .catalog(AWS_MP_CATALOG)
    .entityType(ENTITY_TYPE_RESALE_AUTHORIZATION)
    .maxResults(10)
    .ownershipType(OWNERSHIP_TYPE_SHARED)
    .nextToken(null)
    .build();

  ListEntitiesResponse listEntitiesResponse =
 marketplaceCatalogClient.listEntities(listEntitiesRequest);

  responseList.add(listEntitiesResponse);

  while (listEntitiesResponse.nextToken() != null) {
   listEntitiesRequest = ListEntitiesRequest.builder()
     .catalog(AWS_MP_CATALOG)
     .entityType(ENTITY_TYPE_RESALE_AUTHORIZATION)
     .maxResults(10)
     .ownershipType(OWNERSHIP_TYPE_SHARED)
     .nextToken(listEntitiesResponse.nextToken())
     .build();

   listEntitiesResponse =
 marketplaceCatalogClient.listEntities(listEntitiesRequest);

   responseList.add(listEntitiesResponse);
  }
  return responseList;
 }

}
```

- Per i dettagli sull'API, consulta la sezione ListEntities AWS SDK for Java 2.xAPI Reference.

Pubblica un CPPO e aggiungi un EULA per l'acquirente

Il seguente esempio di codice mostra come pubblicare un CPPO e aggiungere un EULA per l'acquirente.

SDK per Java 2.x

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri come configurarlo ed eseguirlo nel repository Marketplace AWS API Reference Code Library.

Per eseguire questo esempio, passa il seguente changeset JSON a **`RunChangesets`** in Utilities per avviare un changeset dalla sezione Utilities.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType" : "CreateOfferUsingResaleAuthorization",
            "Entity": {
            "Type": "Offer@1.0"
            },
            "ChangeName": "CreateCPPOoffer",
            "DetailsDocument": {
                "ResaleAuthorizationId":"resaleauthz-1111111111111",
                "Name": "Test Offer",
                "Description":"Test product"
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateCPPOoffer.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
           {
               "Type": "LegalTerm",
               "Documents": [
```

```
                {
                 "Type": "CustomEula",
                 "Url": "https://s3.amazonaws.com/sample-bucket/custom-eula.pdf"
                }
               ]
           }
       ]
         }
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateCPPOoffer.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": ["222222222222"]
                    }
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateCPPOoffer.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-07-31"
                }
        },
        {
            "ChangeType": "UpdateValidityTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateCPPOoffer.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "ValidityTerm",
                        "AgreementDuration": "P450D"
                    }
                ]
```

```
            }
        },
        {
            "ChangeType":"ReleaseOffer",
            "Entity":{
                "Type": "Offer@1.0",
                "Identifier": "$CreateCPPOoffer.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

- Per i dettagli sull'API, consulta API Reference. StartChangeSetAWS SDK for Java 2.x

Pubblica un CPPO utilizzando l'autorizzazione di rivendita una tantum e aggiorna il markup dei prezzi

Il seguente esempio di codice mostra come pubblicare un CPPO utilizzando l'autorizzazione di rivendita una tantum su prodotti AMI, SaaS o Container e aggiornare il markup dei prezzi.

SDK per Java 2.x

> **Note**
>
> C'è di più su. GitHub Trova l'esempio completo e scopri come configurarlo ed eseguirlo nel repository Marketplace AWS API Reference Code Library.

Per eseguire questo esempio, passa il seguente changeset JSON a **`RunChangesets`** in Utilities per avviare un changeset dalla sezione Utilities.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType" : "CreateOfferUsingResaleAuthorization",
            "Entity": {
            "Type": "Offer@1.0"
            },
            "ChangeName": "CreateCPPOoffer",
            "DetailsDocument": {
                "ResaleAuthorizationId":"resaleauthz-1111111111111",
```

```
            "Name": "Test Offer",
            "Description":"Test product"
        }
    },
    {
        "ChangeType": "UpdateMarkup",
        "Entity": {
            "Type": "Offer@1.0",
            "Identifier": "$CreateCPPOoffer.Entity.Identifier"
        },
        "DetailsDocument": {
            "Percentage" : "5.0"
    }
    },
    {
        "ChangeType": "UpdateTargeting",
        "Entity": {
            "Type": "Offer@1.0",
            "Identifier": "$CreateCPPOoffer.Entity.Identifier"
        },
        "DetailsDocument": {
            "PositiveTargeting": {
                "BuyerAccounts": ["222222222222"]
                }
        }
    },
    {
        "ChangeType": "UpdateAvailability",
        "Entity": {
            "Type": "Offer@1.0",
            "Identifier": "$CreateCPPOoffer.Entity.Identifier"
        },
        "DetailsDocument": {
            "AvailabilityEndDate": "2023-07-31"
            }
    },
    {
        "ChangeType": "UpdateValidityTerms",
        "Entity": {
            "Type": "Offer@1.0",
            "Identifier": "$CreateCPPOoffer.Entity.Identifier"
        },
        "DetailsDocument": {
            "Terms": [
```

```
                    {
                        "Type": "ValidityTerm",
                        "AgreementDuration": "P450D"
                    }
                ]
             }
        },
        {
            "ChangeType":"ReleaseOffer",
            "Entity":{
                "Type": "Offer@1.0",
                "Identifier": "$CreateCPPOoffer.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

- Per i dettagli sull'API, consulta API Reference. StartChangeSetAWS SDK for Java 2.x

Pubblica una bozza di CPPO e aggiorna il markup dei prezzi

Il seguente esempio di codice mostra come pubblicare una bozza di CPPO e aggiornare il markup dei prezzi.

SDK per Java 2.x

> **Note**
>
> C'è di più su. GitHub Trova l'esempio completo e scopri come configurarlo ed eseguirlo nel repository Marketplace AWS API Reference Code Library.

Per eseguire questo esempio, passa il seguente changeset JSON a **`RunChangesets`** in Utilities per avviare un changeset dalla sezione Utilities.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType" : "CreateOfferUsingResaleAuthorization",
            "Entity": {
```

```
            "Type": "Offer@1.0"
            },
            "ChangeName": "CreateCPPOoffer",
            "DetailsDocument": {
                "ResaleAuthorizationId":"resaleauthz-1111111111111",
                "Name": "Test Offer",
                "Description":"Test product"
            }
        },
        {
            "ChangeType": "UpdateMarkup",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateCPPOoffer.Entity.Identifier"
            },
            "DetailsDocument": {
                "Percentage" : "5.0"
        }
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateCPPOoffer.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": ["222222222222"]
                    }
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateCPPOoffer.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-07-31"
                }
        },
        {
            "ChangeType": "UpdateValidityTerms",
            "Entity": {
```

```
                "Type": "Offer@1.0",
                "Identifier": "$CreateCPPOoffer.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "ValidityTerm",
                        "AgreementDuration": "P450D"
                    }
                ]
             }
        },
        {
            "ChangeType":"ReleaseOffer",
            "Entity":{
                "Type": "Offer@1.0",
                "Identifier": "$CreateCPPOoffer.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

- Per i dettagli sull'API, consulta API Reference. StartChangeSetAWS SDK for Java 2.x

Aggiorna la data di scadenza di un CPPO

Il seguente esempio di codice mostra come aggiornare la data di scadenza di un CPPO per dare agli acquirenti più tempo per valutare e accettare l'offerta.

SDK per Java 2.x

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri come configurarlo ed eseguirlo nel repository Marketplace AWS API Reference Code Library.

Per eseguire questo esempio, passa il seguente changeset JSON a **`RunChangesets`** in Utilities per avviare un changeset dalla sezione Utilities.

```
{
```

```
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "offer-1111111111111"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2025-07-31"
                }
        }
    ]
}
```

- Per i dettagli sull'API, consulta API Reference. StartChangeSetAWS SDK for Java 2.x

Prodotti container

Crea una bozza di prodotto contenitore con una bozza di offerta pubblica

Il seguente esempio di codice mostra come creare una bozza di prodotto contenitore con una bozza di offerta pubblica.

SDK per Java 2.x

> **Note**
>
> C'è di più su GitHub. Trova l'esempio completo e scopri come configurarlo ed eseguirlo nel repository Marketplace AWS API Reference Code Library.

Per eseguire questo esempio, passa il seguente changeset JSON a **`RunChangesets`** in Utilities per avviare un changeset dalla sezione Utilities.

```
{
    "Catalog":"AWSMarketplace",
    "changeSet":[
        {
            "ChangeType": "CreateProduct",
            "ChangeName": "CreateProductChange",
```

```
            "Entity": {
                "Type": "ContainerProduct@1.0"
            },
            "DetailsDocument": {
                "ProductTitle": "Sample product"
            }
        },
        {
            "ChangeType": "CreateOffer",
            "ChangeName": "CreateOfferChange",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "DetailsDocument": {
                "ProductId": "$CreateProductChange.Entity.Identifier",
                "Name": "Test Offer"
            }
        }
    ]
}
```

- Per i dettagli sull'API, consulta API Reference. StartChangeSetAWS SDK for Java 2.x

Crea un prodotto container limitato con un'offerta pubblica e prezzi contrattuali

Il seguente esempio di codice mostra come creare un prodotto contenitore limitato con un'offerta pubblica, prezzi contrattuali e un EULA standard.

SDK per Java 2.x

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri come configurarlo ed eseguirlo nel repository Marketplace AWS API Reference Code Library.

Per eseguire questo esempio, passa il seguente changeset JSON a **`RunChangesets`** in Utilities per avviare un changeset dalla sezione Utilities.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
```

```
        {
            "ChangeType": "CreateProduct",
            "Entity": {
                "Type": "ContainerProduct@1.0"
            },
            "DetailsDocument": {},
            "ChangeName": "CreateProductChange"
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "ContainerProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "LogoUrl": "https://s3.amazonaws.com/logos/sample.png",
                "Categories": [
                    "Streaming solutions"
                ],
                "ProductTitle": "ContainerProduct",
                "AdditionalResources": [],
                "LongDescription": "Long description goes here",
                "SearchKeywords": [
                    "container streaming"
                ],
                "ShortDescription": "Description1",
                "Highlights": [
                    "Highlight 1",
                    "Highlight 2"
                ],
                "SupportDescription": "No support available",
                "VideoUrls": []
            }
        },
        {
            "ChangeType": "AddDimensions",
            "Entity": {
                "Type": "ContainerProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": [
                {
                    "Key": "Cores",
                    "Description": "Cores per cluster",
```

```
                "Name": "Cores",
                "Types": [
                    "Entitled"
                ],
                "Unit": "Units"
            }
        ]
    },
    {
        "ChangeType": "UpdateTargeting",
        "Entity": {
            "Type": "ContainerProduct@1.0",
            "Identifier": "$CreateProductChange.Entity.Identifier"
        },
        "DetailsDocument": {
            "PositiveTargeting": {
                "BuyerAccounts": [
                    "111111111111"
                ]
            }
        }
    },
    {
        "ChangeType": "AddRepositories",
        "Entity": {
            "Type": "ContainerProduct@1.0",
            "Identifier": "$CreateProductChange.Entity.Identifier"
        },
        "DetailsDocument": {
            "Repositories": [
                {
                    "RepositoryName": "uniquerepositoryname",
                    "RepositoryType": "ECR"
                }
            ]
        }
    },
    {
        "ChangeType": "ReleaseProduct",
        "Entity": {
            "Type": "ContainerProduct@1.0",
            "Identifier": "$CreateProductChange.Entity.Identifier"
        },
        "DetailsDocument": {}
```

```
        },
        {
            "ChangeType": "CreateOffer",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "DetailsDocument": {
                "ProductId": "$CreateProductChange.Entity.Identifier"
            },
            "ChangeName": "CreateOfferChange"
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "Constraints": {
                                    "MultipleDimensionSelection": "Disallowed",
                                    "QuantityConfiguration": "Disallowed"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "Cores",
                                        "Price": "0.25"
                                    }
                                ]
                            }
                        ]
                    }
                ]
            }
```

```
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "StandardEula",
                                "Version": "2022-07-14"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateSupportTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "SupportTerm",
                        "RefundPolicy": "No refunds"
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Some container offer Name",
```

```
                "Description": "Some interesting container offer description"
            }
        },
        {
            "ChangeType": "UpdateRenewalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "RenewalTerm"
                    }
                ]
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

- Per i dettagli sull'API, consulta API Reference. StartChangeSetAWS SDK for Java 2.x

Entità

Descrivi tutte le entità in una singola chiamata

Il seguente esempio di codice mostra come descrivere tutte le entità in una singola chiamata.

SDK per Java 2.x

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri come configurarlo ed eseguirlo nel repository Marketplace AWS API Reference Code Library.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package com.example.awsmarketplace.catalogapi;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.*;
import com.example.awsmarketplace.utils.ReferenceCodesUtils;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.services.marketplacecatalog.MarketplaceCatalogClient;
import
 software.amazon.awssdk.services.marketplacecatalog.model.BatchDescribeEntitiesRequest;
import software.amazon.awssdk.services.marketplacecatalog.model.EntityRequest;
import
 software.amazon.awssdk.services.marketplacecatalog.model.BatchDescribeEntitiesResponse;
import software.amazon.awssdk.services.marketplacecatalog.model.EntityDetail;
import
 software.amazon.awssdk.services.marketplacecatalog.model.BatchDescribeErrorDetail;

import java.util.Arrays;
import java.util.Map;

public class BatchDescribeEntities {

    /*
     * BatchDescribe my entities in a single call and
     *  check if it contains all the information I need to know about the entities.
     */
    public static void main(String[] args) {

        MarketplaceCatalogClient marketplaceCatalogClient =
                MarketplaceCatalogClient.builder()
                        .httpClient(ApacheHttpClient.builder().build())
                        .credentialsProvider(ProfileCredentialsProvider.create())
```

```
                        .build();

        BatchDescribeEntitiesRequest batchDescribeEntitiesRequest =
                BatchDescribeEntitiesRequest.builder()
                        .entityRequestList(Arrays.asList(
                                EntityRequest.builder()
                                        .catalog(AWS_MP_CATALOG).entityId(OFFER_ID)
                                        .build(),
                                EntityRequest.builder()

 .catalog(AWS_MP_CATALOG).entityId(PRODUCT_ID)
                                        .build()))
                        .build();

        BatchDescribeEntitiesResponse batchDescribeEntitiesResponse =
 marketplaceCatalogClient.batchDescribeEntities(batchDescribeEntitiesRequest);

        // Reading the successful entities response
        Map<String, EntityDetail> entityDetailsMap =
 batchDescribeEntitiesResponse.entityDetails();
        for (Map.Entry<String, EntityDetail> entry : entityDetailsMap.entrySet()) {
            System.out.println("EntityId: " + entry.getKey());
            ReferenceCodesUtils.formatOutput(entry.getValue());
        }

        // Logging the failed entities error details
        Map<String, BatchDescribeErrorDetail> entityErrorsMap =
 batchDescribeEntitiesResponse.errors();
        for (Map.Entry<String, BatchDescribeErrorDetail> entry :
 entityErrorsMap.entrySet()) {
            System.out.println(String.format("EntityId: %s, ErrorCode: %s,
 ErrorMessage: %s", entry.getKey(),
                    entry.getValue().errorCode(), entry.getValue().errorMessage()));
        }
    }
}
```

- Per i dettagli sull'API, consulta la sezione BatchDescribeEntities AWS SDK for Java 2.xAPI Reference.

Elenca e descrivi tutte le offerte associate a un prodotto

Il seguente esempio di codice mostra come elencare e descrivere tutte le offerte associate a un prodotto.

SDK per Java 2.x

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri come configurarlo ed eseguirlo nel repository Marketplace AWS API Reference Code Library.

```
package com.example.awsmarketplace.catalogapi;

import java.util.ArrayList;
import java.util.List;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.*;
import com.example.awsmarketplace.utils.ReferenceCodesUtils;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.services.marketplacecatalog.MarketplaceCatalogClient;
import
 software.amazon.awssdk.services.marketplacecatalog.model.DescribeEntityRequest;
import
 software.amazon.awssdk.services.marketplacecatalog.model.DescribeEntityResponse;
import software.amazon.awssdk.services.marketplacecatalog.model.EntitySummary;
import software.amazon.awssdk.services.marketplacecatalog.model.EntityTypeFilters;
import software.amazon.awssdk.services.marketplacecatalog.model.ListEntitiesRequest;
import
 software.amazon.awssdk.services.marketplacecatalog.model.ListEntitiesResponse;
import software.amazon.awssdk.services.marketplacecatalog.model.OfferFilters;
import
 software.amazon.awssdk.services.marketplacecatalog.model.OfferProductIdFilter;
import
 software.amazon.awssdk.services.marketplacecatalog.model.OfferTargetingFilter;

public class ListProductPrivateOffers {

 private static MarketplaceCatalogClient marketplaceCatalogClient =
```

```
   MarketplaceCatalogClient.builder()
   .httpClient(ApacheHttpClient.builder().build())
   .credentialsProvider(ProfileCredentialsProvider.create())
   .build();
/*
 * retrieve all private offer information related to a single product
 */
public static void main(String[] args) {

 List<EntitySummary> entitySummaryList = getEntitySummaryList();

 // for each offer id, output the offer detail using DescribeEntity API

 for (EntitySummary entitySummary : entitySummaryList) {
  DescribeEntityRequest describeEntityRequest =
    DescribeEntityRequest.builder()
    .catalog(AWS_MP_CATALOG)
    .entityId(entitySummary.entityId())
    .build();
  DescribeEntityResponse describeEntityResponse =
marketplaceCatalogClient.describeEntity(describeEntityRequest);
  ReferenceCodesUtils.formatOutput(describeEntityResponse);
 }
}
public static List<EntitySummary> getEntitySummaryList() {
 // define list entities filters

 EntityTypeFilters entityTypeFilters =
   EntityTypeFilters.builder()
   .offerFilters(OfferFilters.builder()
     .targeting(OfferTargetingFilter.builder()
       .valueListWithStrings(OFFER_TARGETING_BUYERACCOUNTS)
       .build())
     .productId(OfferProductIdFilter.builder()
       .valueList(PRODUCT_ID)
       .build())
     .build())
   .build();

 ListEntitiesRequest listEntitiesRequest =
   ListEntitiesRequest.builder()
   .catalog(AWS_MP_CATALOG)
   .entityType(ENTITY_TYPE_OFFER).maxResults(50)
   .entityTypeFilters(entityTypeFilters)
```

```
    .nextToken(null)
    .build();

 ListEntitiesResponse listEntitiesResponse =
marketplaceCatalogClient.listEntities(listEntitiesRequest);

 // save all entitySummary of the results into entitySummaryList

 List<EntitySummary> entitySummaryList = new ArrayList<EntitySummary>();

 entitySummaryList.addAll(listEntitiesResponse.entitySummaryList());

 while ( listEntitiesResponse.nextToken() != null &&
listEntitiesResponse.nextToken().length() > 0) {
  listEntitiesRequest =
    ListEntitiesRequest.builder()
    .catalog(AWS_MP_CATALOG)
    .entityType(ENTITY_TYPE_OFFER).maxResults(50)
    .entityTypeFilters(entityTypeFilters)
    .nextToken(listEntitiesResponse.nextToken())
    .build();
  listEntitiesResponse =
marketplaceCatalogClient.listEntities(listEntitiesRequest);
  entitySummaryList.addAll(listEntitiesResponse.entitySummaryList());
 }
 return entitySummaryList;
}

}
```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella Documentazione di riferimento delle API AWS SDK for Java 2.x .
  - DescribeEntity
  - ListEntities

Offers (Offerte)

Crea una dimensione personalizzata per un prodotto SaaS e crea un'offerta privata

L'esempio di codice seguente mostra come creare una dimensione personalizzata per un prodotto SaaS e creare un'offerta privata.

SDK per Java 2.x

> **Note**
>
> C'è di più su. GitHub Trova l'esempio completo e scopri come configurarlo ed eseguirlo nel repository Marketplace AWS API Reference Code Library.

Per eseguire questo esempio, passa il seguente changeset JSON a **`RunChangesets`** in Utilities per avviare un changeset dalla sezione Utilities.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "AddDimensions",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "prod-1111111111111"
            },
            "DetailsDocument": [
                {
                    "Types": [
                        "Entitled"
                    ],
                    "Description": "Custom Pricing 4 w/ terms and coverage to be
 defined in Private Offer",
                    "Unit": "Units",
                    "Key": "Custom4",
                    "Name": "Custom Pricing 4"
                }
            ]
        },
        {
            "ChangeType": "CreateOffer",
```

```
            "Entity": {
                "Type": "Offer@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111"
            },
            "ChangeName": "CreateOfferChange"
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Private Test Offer - SaaS Contract Product",
                "Description": "Private Test Offer - SaaS Contract Product"
            }
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111"
                    ]
                }
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
```

```
                        {
                            "Type": "StandardEula",
                            "Version": "2022-07-14"
                        }
                    ]
                }
            ]
        }
    },
    {
        "ChangeType": "UpdateAvailability",
        "Entity": {
            "Type": "Offer@1.0",
            "Identifier": "$CreateOfferChange.Entity.Identifier"
        },
        "DetailsDocument": {
            "AvailabilityEndDate": "2023-12-31"
        }
    },
    {
        "ChangeType": "UpdatePricingTerms",
        "Entity": {
            "Type": "Offer@1.0",
            "Identifier": "$CreateOfferChange.Entity.Identifier"
        },
        "DetailsDocument": {
            "PricingModel": "Contract",
            "Terms": [
                {
                    "Type": "ConfigurableUpfrontPricingTerm",
                    "CurrencyCode": "USD",
                    "RateCards": [
                        {
                            "Constraints": {
                                "MultipleDimensionSelection": "Allowed",
                                "QuantityConfiguration": "Allowed"
                            },
                            "RateCard": [
                                {
                                    "DimensionKey": "Custom4",
                                    "Price": "300.0"
                                }
                            ],
                            "Selector": {
```

```
                                "Type": "Duration",
                                "Value": "P36M"
                            }
                        }
                    ]
                }
            ]
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ],
    "ChangeSetName": "PrivateOfferWithCustomDimension"
}
```

- Per i dettagli sull'API, consulta API Reference. StartChangeSetAWS SDK for Java 2.x

Creare una bozza di offerta privata per un prodotto AMI o SaaS

Il seguente esempio di codice mostra come creare una bozza di offerta privata per un prodotto AMI o SaaS in modo da poterla esaminare internamente prima di pubblicarla agli acquirenti.

SDK per Java 2.x

> **Note**
>
> C'è di più su. GitHub Trova l'esempio completo e scopri come configurarlo ed eseguirlo nel repository Marketplace AWS API Reference Code Library.

Per eseguire questo esempio, passa il seguente changeset JSON a **`RunChangesets`** in Utilities per avviare un changeset dalla sezione Utilities.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
```

```
        {
            "ChangeType": "CreateOffer",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "Test Private Offer"
            }
        }
    ]
}
```

- Per i dettagli sull'API, consulta API Reference. StartChangeSetAWS SDK for Java 2.x

Crea un'offerta privata con contratto e Pay-As-You-Go prezzi per un prodotto SaaS

Il seguente esempio di codice mostra come creare un'offerta privata con contratto e Pay-As-You-Go prezzi per un prodotto SaaS.

SDK per Java 2.x

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri come configurarlo ed eseguirlo nel repository Marketplace AWS API Reference Code Library.

Per eseguire questo esempio, passa il seguente changeset JSON a **`RunChangesets`** in Utilities per avviare un changeset dalla sezione Utilities.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateOffer",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "ChangeName": "CreateOfferChange",
            "DetailsDocument": {
                "ProductId": "prod-1111111111111"
```

```
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test private offer for SaaSProduct using AWS Marketplace
API Reference Code",
                "Description": "Test private offer with subscription pricing for
SaaSProduct using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111",
                        "222222222222"
                    ]
                }
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "UsageBasedPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
```

```
                            "RateCard": [
                                {
                                    "DimensionKey": "WorkloadSmall",
                                    "Price": "0.15"
                                },
                                {
                                    "DimensionKey": "WorkloadMedium",
                                    "Price": "0.25"
                                }
                            ]
                        }
                    ]
                },
                {
                    "Type": "ConfigurableUpfrontPricingTerm",
                    "CurrencyCode": "USD",
                    "RateCards": [
                        {
                            "Selector": {
                                "Type": "Duration",
                                "Value": "P12M"
                            },
                            "RateCard": [
                                {
                                    "DimensionKey": "BasicService",
                                    "Price": "150"
                                },
                                {
                                    "DimensionKey": "PremiumService",
                                    "Price": "300"
                                }
                            ],
                            "Constraints": {
                                "MultipleDimensionSelection": "Allowed",
                                "QuantityConfiguration": "Allowed"
                            }
                        }
                    ]
                }
            ]
        }
    },
    {
        "ChangeType": "UpdateLegalTerms",
```

```
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-12-31"
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

- Per i dettagli sull'API, consulta API Reference. StartChangeSetAWS SDK for Java 2.x

Crea un'offerta privata con prezzi contrattuali e un programma di pagamento flessibile per un prodotto SaaS

Il seguente esempio di codice mostra come creare un'offerta privata con prezzi contrattuali e un piano di pagamento flessibile per un prodotto SaaS.

SDK per Java 2.x

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri come configurarlo ed eseguirlo nel repository Marketplace AWS API Reference Code Library.

Per eseguire questo esempio, passa il seguente changeset JSON a **`RunChangesets`** in Utilities per avviare un changeset dalla sezione Utilities.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateOffer",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "ChangeName": "CreateOfferChange",
            "DetailsDocument": {
                "ProductId": "prod-1111111111111"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test private offer for SaaSProduct using AWS Marketplace
 API Reference Code",
                "Description": "Test private offer with subscription pricing for
 SaaSProduct using AWS Marketplace API Reference Code"
            }
```

```
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111",
                        "222222222222"
                    ]
                }
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "FixedUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "Price": "0.0",
                        "Grants": [
                            {
                                "DimensionKey": "BasicService",
                                "MaxQuantity": 1
                            },
                            {
                                "DimensionKey": "PremiumService",
                                "MaxQuantity": 1
                            }
                        ]
                    }
                ]
            }
        },
        {
```

```
            "ChangeType": "UpdateValidityTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "ValidityTerm",
                        "AgreementDuration": "P12M"
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdatePaymentScheduleTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "PaymentScheduleTerm",
                        "CurrencyCode": "USD",
                        "Schedule": [
                            {
                                "ChargeDate": "2024-01-01",
                                "ChargeAmount": "200.00"
                            },
                            {
                                "ChargeDate": "2024-02-01",
                                "ChargeAmount": "170.00"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
```

```
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-12-31"
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

- Per i dettagli sull'API, consulta API Reference. StartChangeSetAWS SDK for Java 2.x

Crea un'offerta privata con prezzi contrattuali per un prodotto Container

Il seguente esempio di codice mostra come creare un'offerta privata con prezzi contrattuali per un prodotto Container.

SDK per Java 2.x

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri come configurarlo ed eseguirlo nel repository Marketplace AWS API Reference Code Library.

Per eseguire questo esempio, passa il seguente changeset JSON a **`RunChangesets`** in Utilities per avviare un changeset dalla sezione Utilities.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateOffer",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "ChangeName": "CreateOfferChange",
            "DetailsDocument": {
                "ProductId": "prod-1111111111111"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test private offer for Container product using AWS
 Marketplace API Reference Code",
                "Description": "Test private offer for Container product with
 contract pricing using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
```

```
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111"
                    ]
                }
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "Constraints": {
                                    "MultipleDimensionSelection": "Disallowed",
                                    "QuantityConfiguration": "Disallowed"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "ReqPerHour",
                                        "Price": "0.25"
                                    }
                                ]
                            }
                        ]
                    }
                ]
            }
        },
        {
```

```
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "StandardEula",
                                "Version": "2022-07-14"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-12-31"
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

- Per i dettagli sull'API, consulta API Reference. StartChangeSetAWS SDK for Java 2.x

Crea un'offerta privata con prezzi contrattuali per un prodotto AMI

Il seguente esempio di codice mostra come creare un'offerta privata con prezzi contrattuali per un prodotto AMI.

SDK per Java 2.x

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri come configurarlo ed eseguirlo nel repository Marketplace AWS API Reference Code Library.

Per eseguire questo esempio, passa il seguente changeset JSON a **`RunChangesets`** in Utilities per avviare un changeset dalla sezione Utilities.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateOffer",
            "ChangeName": "CreateOfferChange",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test private offer for AmiProduct using AWS Marketplace API
 Reference Code",
                "Description": "Test private offer with hourly annual pricing for
 AmiProduct using AWS Marketplace API Reference Code"
            }
        },
```

```
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111",
                        "222222222222"
                    ]
                }
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-12-31"
```

```
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "ReadOnlyUsers",
                                        "Price": "220.00"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
```

```
    ]
}
```

- Per i dettagli sull'API, consulta API Reference. StartChangeSetAWS SDK for Java 2.x

Crea un'offerta privata con prezzi orari annuali e un programma di pagamento flessibile per un prodotto AMI

Il seguente esempio di codice mostra come creare un'offerta privata con prezzi orari annuali e un programma di pagamento flessibile per un prodotto AMI.

SDK per Java 2.x

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri come configurarlo ed eseguirlo nel repository Marketplace AWS API Reference Code Library.

Per eseguire questo esempio, passa il seguente changeset JSON a **`RunChangesets`** in Utilities per avviare un changeset dalla sezione Utilities.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateOffer",
            "ChangeName": "CreateOfferChange",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
```

```
            "DetailsDocument": {
                "Name": "Test private offer for AmiProduct using AWS Marketplace API
 Reference Code",
                "Description": "Test private offer with hourly annual pricing for
 AmiProduct using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111",
                        "222222222222"
                    ]
                }
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        },
        {
```

```
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-12-31"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Usage",
                "Terms": [
                    {
                        "Type": "UsageBasedPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.micro",
                                        "Price": "0.17"
                                    }
                                ]
                            }
                        ]
                    },
                    {
                        "Type": "FixedUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "Price": "0.0",
                        "Duration": "P365D",
                        "Grants": [
                            {
                                "DimensionKey": "t2.micro",
                                "MaxQuantity": 1
                            }
                        ]
                    }
```

```
                ]
            }
        },
        {
            "ChangeType": "UpdateValidityTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "ValidityTerm",
                        "AgreementDuration": "P650D"
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdatePaymentScheduleTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "PaymentScheduleTerm",
                        "CurrencyCode": "USD",
                        "Schedule": [
                            {
                                "ChargeDate": "2024-01-01",
                                "ChargeAmount": "200.00"
                            },
                            {
                                "ChargeDate": "2024-02-01",
                                "ChargeAmount": "170.00"
                            }
                        ]
                    }
                ]
            }
        },
        {
```

```
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

- Per i dettagli sull'API, consulta API Reference. StartChangeSetAWS SDK for Java 2.x

Crea un'offerta privata con prezzi orari annuali per un prodotto AMI

Il seguente esempio di codice mostra come creare un'offerta privata con prezzi orari annuali per un prodotto AMI.

SDK per Java 2.x

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri come configurarlo ed eseguirlo nel repository Marketplace AWS API Reference Code Library.

Per eseguire questo esempio, passa il seguente changeset JSON a **`RunChangesets`** in Utilities per avviare un changeset dalla sezione Utilities.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateOffer",
            "ChangeName": "CreateOfferChange",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111"
            }
        },
        {
```

```
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test private offer for AmiProduct using AWS Marketplace API
 Reference Code",
                "Description": "Test private offer with hourly annual pricing for
 AmiProduct using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111",
                        "222222222222"
                    ]
                }
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
```

```
                }
            ]
        }
    },
    {
        "ChangeType": "UpdateAvailability",
        "Entity": {
            "Type": "Offer@1.0",
            "Identifier": "$CreateOfferChange.Entity.Identifier"
        },
        "DetailsDocument": {
            "AvailabilityEndDate": "2023-12-31"
        }
    },
    {
        "ChangeType": "UpdatePricingTerms",
        "Entity": {
            "Type": "Offer@1.0",
            "Identifier": "$CreateOfferChange.Entity.Identifier"
        },
        "DetailsDocument": {
            "PricingModel": "Usage",
            "Terms": [
                {
                    "Type": "UsageBasedPricingTerm",
                    "CurrencyCode": "USD",
                    "RateCards": [
                        {
                            "RateCard": [
                                {
                                    "DimensionKey": "t2.micro",
                                    "Price": "0.17"
                                }
                            ]
                        }
                    ]
                },
                {
                    "Type": "ConfigurableUpfrontPricingTerm",
                    "CurrencyCode": "USD",
                    "RateCards": [
                        {
                            "Selector": {
                                "Type": "Duration",
```

```
                                "Value": "P365D"
                            },
                            "RateCard": [
                                {
                                    "DimensionKey": "t2.micro",
                                    "Price": "220.00"
                                }
                            ],
                            "Constraints": {
                                "MultipleDimensionSelection": "Allowed",
                                "QuantityConfiguration": "Allowed"
                            }
                        }
                    ]
                }
            ]
        }
    },
    {
        "ChangeType": "UpdateValidityTerms",
        "Entity": {
            "Type": "Offer@1.0",
            "Identifier": "$CreateOfferChange.Entity.Identifier"
        },
        "DetailsDocument": {
            "Terms": [
                {
                    "Type": "ValidityTerm",
                    "AgreementDuration": "P650D"
                }
            ]
        }
    },
    {
        "ChangeType": "ReleaseOffer",
        "Entity": {
            "Type": "Offer@1.0",
            "Identifier": "$CreateOfferChange.Entity.Identifier"
        },
        "DetailsDocument": {}
    }
]
}
```

- Per i dettagli sull'API, consulta API Reference. StartChangeSetAWS SDK for Java 2.x

Crea un'offerta privata con prezzi orari per un prodotto AMI

Il seguente esempio di codice mostra come creare un'offerta privata con prezzi orari per un prodotto AMI.

SDK per Java 2.x

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri come configurarlo ed eseguirlo nel repository Marketplace AWS API Reference Code Library.

Per eseguire questo esempio, passa il seguente changeset JSON a **`RunChangesets`** in Utilities per avviare un changeset dalla sezione Utilities.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateOffer",
            "ChangeName": "CreateOfferChange",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test private offer for AmiProduct using AWS Marketplace API
 Reference Code",
                "Description": "Test private offer with hourly pricing for
 AmiProduct using AWS Marketplace API Reference Code"
```

```
            }
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111",
                        "222222222222"
                    ]
                }
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "StandardEula",
                                "Version": "2022-07-14"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
```

```
                "AvailabilityEndDate": "2025-01-01"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Usage",
                "Terms": [
                    {
                        "Type": "UsageBasedPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.micro",
                                        "Price": "0.15"
                                    }
                                ]
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateValidityTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "ValidityTerm",
                        "AgreementDuration": "P30D"
                    }
                ]
            }
        },
```

```
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

- Per i dettagli sull'API, consulta API Reference. StartChangeSetAWS SDK for Java 2.x

Crea un'offerta privata con prezzi di abbonamento per un prodotto SaaS

Il seguente esempio di codice mostra come creare un'offerta privata con prezzi di abbonamento per un prodotto SaaS.

SDK per Java 2.x

> **Note**
>
> C'è di più su. GitHub Trova l'esempio completo e scopri come configurarlo ed eseguirlo nel repository Marketplace AWS API Reference Code Library.

Per eseguire questo esempio, passa il seguente changeset JSON a **`RunChangesets`** in Utilities per avviare un changeset dalla sezione Utilities.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateOffer",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "ChangeName": "CreateOfferChange",
            "DetailsDocument": {
                "ProductId": "prod-1111111111111"
            }
        },
```

```
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test private offer for SaaSProduct using AWS Marketplace
API Reference Code",
                "Description": "Test private offer with subscription pricing for
SaaSProduct using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111",
                        "222222222222"
                    ]
                }
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Usage",
                "Terms": [
                    {
                        "Type": "UsageBasedPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "RateCard": [
                                    {
```

```
                                    "DimensionKey": "WorkloadSmall",
                                    "Price": "0.13"
                                },
                                {
                                    "DimensionKey": "WorkloadMedium",
                                    "Price": "0.22"
                                }
                            ]
                        }
                    ]
                }
            ]
        },
        {
            "ChangeType": "UpdateValidityTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "ValidityTerm",
                        "AgreementDuration": "P30D"
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
```

```
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-12-31"
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

- Per i dettagli sull'API, consulta API Reference. StartChangeSetAWS SDK for Java 2.x

Crea un'offerta privata con prezzi contrattuali a più livelli per un prodotto SaaS

Il seguente esempio di codice mostra come creare un'offerta privata con prezzi contrattuali a più livelli per un prodotto SaaS.

SDK per Java 2.x

> **Note**
>
> C'è di più su. GitHub Trova l'esempio completo e scopri come configurarlo ed eseguirlo nel repository Marketplace AWS API Reference Code Library.

Per eseguire questo esempio, passa il seguente changeset JSON a **`RunChangesets`** in Utilities per avviare un changeset dalla sezione Utilities.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateOffer",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "ChangeName": "CreateOfferChange",
            "DetailsDocument": {
                "ProductId": "prod-1111111111111"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test private offer for SaaSProduct using AWS Marketplace
 API Reference Code",
                "Description": "Test private offer with subscription pricing for
 SaaSProduct using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111",
                        "222222222222"
                    ]
                }
            }
```

```
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "BasicService",
                                        "Price": "120.00"
                                    },
                                    {
                                        "DimensionKey": "PremiumService",
                                        "Price": "200.00"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Disallowed",
                                    "QuantityConfiguration": "Disallowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
```

```
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-12-31"
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

- Per i dettagli sull'API, consulta API Reference. StartChangeSetAWS SDK for Java 2.x

Crea un'offerta di prova gratuita pubblica con prezzi di abbonamento per un prodotto SaaS

Il seguente esempio di codice mostra come creare un'offerta di prova gratuita pubblica con prezzi di abbonamento per un prodotto SaaS.

SDK per Java 2.x

> **Note**
>
> C'è di più su. GitHub Trova l'esempio completo e scopri come configurarlo ed eseguirlo nel repository Marketplace AWS API Reference Code Library.

Per eseguire questo esempio, passa il seguente changeset JSON a **`RunChangesets`** in Utilities per avviare un changeset dalla sezione Utilities.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateOffer",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "ChangeName": "CreateOfferChange",
            "DetailsDocument": {
                "ProductId": "prod-1111111111111"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test public free trial offer for SaaSProduct using AWS
 Marketplace API Reference Code",
                "Description": "Test public free trial offer with subscription
 pricing for SaaSProduct using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
```

```
            },
            "DetailsDocument": {
                "PricingModel": "Free",
                "Terms": [
                    {
                        "Type": "FreeTrialPricingTerm",
                        "Duration": "P20D",
                        "Grants": [
                            {
                                "DimensionKey": "WorkloadSmall"
                            },
                            {
                                "DimensionKey": "WorkloadMedium"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "StandardEula",
                                "Version": "2022-07-14"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
```

```
            },
            "DetailsDocument": {}
        }
    ]
}
```

- Per i dettagli sull'API, consulta API Reference. StartChangeSetAWS SDK for Java 2.x

Crea un'offerta privata sostitutiva con prezzi contrattuali

Il seguente esempio di codice mostra come creare un'offerta privata sostitutiva a partire da un accordo esistente con prezzi contrattuali.

SDK per Java 2.x

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri come configurarlo ed eseguirlo nel repository Marketplace AWS API Reference Code Library.

Per eseguire questo esempio, passa il seguente changeset JSON a **`RunChangesets`** in Utilities per avviare un changeset dalla sezione Utilities.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType" : "CreateReplacementOffer",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "ChangeName": "CreateReplacementOffer",
            "DetailsDocument": {
                "AgreementId": "agmt-11111111111111111111111"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
```

```
                "Identifier": "$CreateReplacementOffer.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test replacement offer for SaaSProduct using AWS
Marketplace API Reference Codes",
                "Description": "Test private replacement offer with contract pricing
for SaaSProduct"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateReplacementOffer.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "FixedUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "Price": "0.0",
                        "Grants": [
                            {
                                "DimensionKey": "BasicService",
                                "MaxQuantity": 2
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateValidityTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateReplacementOffer.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "ValidityTerm",
                        "AgreementEndDate": "2024-01-30"
                    }
```

```
                ]
            }
        },
        {
            "ChangeType": "UpdatePaymentScheduleTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateReplacementOffer.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "PaymentScheduleTerm",
                        "CurrencyCode": "USD",
                        "Schedule": [
                            {
                                "ChargeDate": "2024-01-01",
                                "ChargeAmount": "0"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateReplacementOffer.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "StandardEula",
                                "Version": "2022-07-14"
                            }
                        ]
                    }
                ]
            }
        },
```

```
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateReplacementOffer.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-12-31"
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateReplacementOffer.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

- Per i dettagli sull'API, consulta API Reference. StartChangeSetAWS SDK for Java 2.x

Descrivi un'offerta pubblica

Il seguente esempio di codice mostra come descrivere un'offerta pubblica.

SDK per Java 2.x

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri come configurarlo ed eseguirlo nel repository Marketplace AWS API Reference Code Library.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package com.example.awsmarketplace.catalogapi;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.*;
import com.example.awsmarketplace.utils.ReferenceCodesUtils;
```

```
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.services.marketplacecatalog.MarketplaceCatalogClient;
import
 software.amazon.awssdk.services.marketplacecatalog.model.DescribeEntityRequest;
import
 software.amazon.awssdk.services.marketplacecatalog.model.DescribeEntityResponse;

public class DescribeEntity {

 /*
  * Describe my AMI or SaaS or Container product and check if it contains all the
 information I need to know about the product
  */
 public static void main(String[] args) {

  String offerId = args.length > 0 ? args[0] : OFFER_ID;

  DescribeEntityResponse describeEntityResponse =
 getDescribeEntityResponse(offerId);

  ReferenceCodesUtils.formatOutput(describeEntityResponse);
 }

 public static DescribeEntityResponse getDescribeEntityResponse(String offerId) {
  MarketplaceCatalogClient marketplaceCatalogClient =
    MarketplaceCatalogClient.builder()
    .httpClient(ApacheHttpClient.builder().build())
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

  DescribeEntityRequest describeEntityRequest =
    DescribeEntityRequest.builder()
    .catalog(AWS_MP_CATALOG)
    .entityId(offerId)
    .build();

  DescribeEntityResponse describeEntityResponse =
 marketplaceCatalogClient.describeEntity(describeEntityRequest);
  return describeEntityResponse;
 }
}
```

- Per i dettagli sull'API, consulta la sezione DescribeEntity AWS SDK for Java 2.xAPI Reference.

Far scadere una bozza di offerta privata

Il seguente esempio di codice mostra come impostare la data di scadenza di un'offerta privata su una data precedente in modo che gli acquirenti non vedano più l'offerta.

SDK per Java 2.x

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri come configurarlo ed eseguirlo nel repository Marketplace AWS API Reference Code Library.

Per eseguire questo esempio, passa il seguente changeset JSON a **`RunChangesets`** in Utilities per avviare un changeset dalla sezione Utilities.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "offer-1111111111111"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-01-01"
            }
        }
    ]
}
```

- Per i dettagli sull'API, consulta API Reference. StartChangeSetAWS SDK for Java 2.x

Elenca tutte le offerte private

Il seguente esempio di codice mostra come elencare tutte le offerte private.

SDK per Java 2.x

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri come configurarlo ed eseguirlo nel repository Marketplace AWS API Reference Code Library.

```
package com.example.awsmarketplace.catalogapi;

import java.util.ArrayList;
import java.util.List;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.*;
import com.example.awsmarketplace.utils.ReferenceCodesUtils;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.services.marketplacecatalog.MarketplaceCatalogClient;
import
 software.amazon.awssdk.services.marketplacecatalog.model.DescribeEntityRequest;
import
 software.amazon.awssdk.services.marketplacecatalog.model.DescribeEntityResponse;
import software.amazon.awssdk.services.marketplacecatalog.model.EntitySummary;
import software.amazon.awssdk.services.marketplacecatalog.model.EntityTypeFilters;
import software.amazon.awssdk.services.marketplacecatalog.model.ListEntitiesRequest;
import
 software.amazon.awssdk.services.marketplacecatalog.model.ListEntitiesResponse;
import
 software.amazon.awssdk.services.marketplacecatalog.model.OfferAvailabilityEndDateFilter;
import
 software.amazon.awssdk.services.marketplacecatalog.model.OfferAvailabilityEndDateFilterDate
import
 software.amazon.awssdk.services.marketplacecatalog.model.OfferBuyerAccountsFilter;
import software.amazon.awssdk.services.marketplacecatalog.model.OfferFilters;
import
 software.amazon.awssdk.services.marketplacecatalog.model.OfferReleaseDateFilter;
import
 software.amazon.awssdk.services.marketplacecatalog.model.OfferReleaseDateFilterDateRange;
import
 software.amazon.awssdk.services.marketplacecatalog.model.OfferTargetingFilter;
```

```
public class ListAllPrivateOffers {

 /*
  * List all my private offers and sort or filter them by Offer Publish Date, Offer
 Expiry Date and Buyer IDs
  *
  * OfferTargetingFilter = BuyerAccounts (private offer);
  * OfferBuyerAccountsFilter: Buyer IDs filter
  * OfferAvailabilityEndDateFilter : Offer Expiry Date filter
  * OfferReleaseDateFilter : Offer Publish Date filter
  */

 private static MarketplaceCatalogClient marketplaceCatalogClient =
   MarketplaceCatalogClient.builder()
   .httpClient(ApacheHttpClient.builder().build())
   .credentialsProvider(ProfileCredentialsProvider.create())
   .build();

 public static void main(String[] args) {

  String offerReleaseDateAfterValue = "2023-01-01T23:59:59Z";
  String offerAvailableEndDateAfterValue = "2040-12-24T23:59:59Z";

  List<EntitySummary> entitySummaryList =
 getEntitySummaryList(offerReleaseDateAfterValue, offerAvailableEndDateAfterValue);

  // for each offer id, output the offer detail using DescribeEntity API


  for (EntitySummary entitySummary : entitySummaryList) {
   DescribeEntityRequest describeEntityRequest =
     DescribeEntityRequest.builder()
     .catalog(AWS_MP_CATALOG)
     .entityId(entitySummary.entityId())
     .build();
   DescribeEntityResponse describeEntityResponse =
 marketplaceCatalogClient.describeEntity(describeEntityRequest);
   ReferenceCodesUtils.formatOutput(describeEntityResponse);
  }
 }

 public static List<EntitySummary> getEntitySummaryList (String
 offerReleaseDateAfterValue, String offerAvailableEndDateAfterValue) {
```

```
 EntityTypeFilters entityTypeFilters =
   EntityTypeFilters.builder()
   .offerFilters(OfferFilters.builder()
     .targeting(OfferTargetingFilter.builder()
       .valueListWithStrings(OFFER_TARGETING_BUYERACCOUNTS)
       .build())
     .buyerAccounts(OfferBuyerAccountsFilter.builder()
       .wildCardValue(BUYER_ACCOUNT_ID)
       .build())
     .availabilityEndDate(OfferAvailabilityEndDateFilter.builder()
       .dateRange(OfferAvailabilityEndDateFilterDateRange.builder()
         .afterValue(offerAvailableEndDateAfterValue).build())
       .build())
     .releaseDate(OfferReleaseDateFilter.builder()
       .dateRange(OfferReleaseDateFilterDateRange.builder()
         .afterValue(offerReleaseDateAfterValue)
         .build())
       .build())
     .build())
   .build();

 ListEntitiesRequest listEntitiesRequest =
   ListEntitiesRequest.builder()
   .catalog(AWS_MP_CATALOG)
   .entityType(ENTITY_TYPE_OFFER).maxResults(10)
   .entityTypeFilters(entityTypeFilters)
   .nextToken(null)
   .build();

 ListEntitiesResponse listEntitiesResponse =
marketplaceCatalogClient.listEntities(listEntitiesRequest);
 List<EntitySummary> entitySummaryList = new ArrayList<EntitySummary>();

 entitySummaryList.addAll(listEntitiesResponse.entitySummaryList());

 while ( listEntitiesResponse.nextToken() != null &&
listEntitiesResponse.nextToken().length() > 0) {
  listEntitiesRequest =
    ListEntitiesRequest.builder()
    .catalog(AWS_MP_CATALOG)
    .entityType(ENTITY_TYPE_OFFER)
    .maxResults(10)
    .entityTypeFilters(entityTypeFilters)
    .nextToken(listEntitiesResponse.nextToken())
```

```
      .build();
    listEntitiesResponse =
 marketplaceCatalogClient.listEntities(listEntitiesRequest);
    entitySummaryList.addAll(listEntitiesResponse.entitySummaryList());
  }

  return entitySummaryList;
 }

}
```

- Per i dettagli sull'API, consulta la sezione StartChangeSet AWS SDK for Java 2.xAPI Reference.

Elenca le offerte pubbliche e private rilasciate per un ID di prodotto specifico

Il seguente esempio di codice mostra come elencare le offerte pubbliche e private rilasciate per un ID di prodotto specifico.

SDK per Java 2.x

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri come configurarlo ed eseguirlo nel repository Marketplace AWS API Reference Code Library.

```
package com.example.awsmarketplace.catalogapi;

import java.util.ArrayList;
import java.util.List;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.*;
import com.example.awsmarketplace.utils.ReferenceCodesUtils;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.services.marketplacecatalog.MarketplaceCatalogClient;
import software.amazon.awssdk.services.marketplacecatalog.model.EntitySummary;
import software.amazon.awssdk.services.marketplacecatalog.model.EntityTypeFilters;
```

```
import software.amazon.awssdk.services.marketplacecatalog.model.ListEntitiesRequest;
import
 software.amazon.awssdk.services.marketplacecatalog.model.ListEntitiesResponse;
import software.amazon.awssdk.services.marketplacecatalog.model.OfferFilters;
import
 software.amazon.awssdk.services.marketplacecatalog.model.OfferProductIdFilter;
import software.amazon.awssdk.services.marketplacecatalog.model.OfferStateFilter;
import
 software.amazon.awssdk.services.marketplacecatalog.model.OfferTargetingFilter;

public class ListProductPublicOrPrivateReleasedOffers {

 /*
  * List released Public/Private offers for a specific product id.
  * Example below is to list released public offers.
  * To change to released private offers, change OFFER_TARGETING_NONE (None) to
 OFFER_TARGETING_BUYERACCOUNTS(BuyerAccounts)
  */
 public static void main(String[] args) {

  List<EntitySummary> entitySummaryList = getEntitySummaryLIst();
  ReferenceCodesUtils.formatOutput(entitySummaryList);
 }

 public static List<EntitySummary> getEntitySummaryLIst() {
  MarketplaceCatalogClient marketplaceCatalogClient =
    MarketplaceCatalogClient.builder()
    .httpClient(ApacheHttpClient.builder().build())
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

  // define list entities filters

  EntityTypeFilters entityTypeFilters =
    EntityTypeFilters.builder()
    .offerFilters(OfferFilters.builder()
      .targeting(OfferTargetingFilter.builder()
        .valueListWithStrings(OFFER_TARGETING_NONE)
        .build())
      .state(OfferStateFilter.builder()
        .valueListWithStrings(OFFER_STATE_RELEASED)
        .build())
      .productId(OfferProductIdFilter.builder()
        .valueList(PRODUCT_ID)
```

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

  ListEntitiesRequest listEntitiesRequest =
    ListEntitiesRequest.builder()
    .catalog(AWS_MP_CATALOG)
    .entityType(ENTITY_TYPE_OFFER)
    .maxResults(10)
    .entityTypeFilters(entityTypeFilters)
    .nextToken(null)
    .build();

  ListEntitiesResponse listEntitiesResponse =
 marketplaceCatalogClient.listEntities(listEntitiesRequest);

  // save all entitySummary of the results into entitySummaryList

  List<EntitySummary> entitySummaryList = new ArrayList<EntitySummary>();

  entitySummaryList.addAll(listEntitiesResponse.entitySummaryList());

  while ( listEntitiesResponse.nextToken() != null &&
 listEntitiesResponse.nextToken().length() > 0) {
    listEntitiesRequest =
      ListEntitiesRequest.builder()
      .catalog(AWS_MP_CATALOG)
      .entityType(ENTITY_TYPE_OFFER)
      .maxResults(10)
      .entityTypeFilters(entityTypeFilters)
      .nextToken(listEntitiesResponse.nextToken())
      .build();
    listEntitiesResponse =
 marketplaceCatalogClient.listEntities(listEntitiesRequest);
    entitySummaryList.addAll(listEntitiesResponse.entitySummaryList());
  }
  return entitySummaryList;
 }

}
```

- Per i dettagli sull'API, consulta la sezione StartChangeSet AWS SDK for Java 2.xAPI Reference.

Aggiorna un'offerta per applicare un contratto con Pay-As-You-Go prezzi

Il seguente esempio di codice mostra come aggiornare un'offerta per applicare un contratto con Pay-As-You-Go prezzi.

SDK per Java 2.x

> **Note**
>
> C'è altro da sapere GitHub. Trova l'esempio completo e scopri come configurarlo ed eseguirlo nel repository Marketplace AWS API Reference Code Library.

Per eseguire questo esempio, passa il seguente changeset JSON a **`RunChangesets`** in Utilities per avviare un changeset dalla sezione Utilities.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "offer-1111111111111"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "UsageBasedPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "RateCard": [
                                    {
                                        "DimensionKey": "WorkloadSmall",
                                        "Price": "0.15"
                                    },
```

```
                                {
                                    "DimensionKey": "WorkloadMedium",
                                    "Price": "0.25"
                                }
                            ]
                        }
                    ]
                },
                {
                    "Type": "ConfigurableUpfrontPricingTerm",
                    "CurrencyCode": "USD",
                    "RateCards": [
                        {
                            "Selector": {
                                "Type": "Duration",
                                "Value": "P12M"
                            },
                            "RateCard": [
                                {
                                    "DimensionKey": "BasicService",
                                    "Price": "150"
                                },
                                {
                                    "DimensionKey": "PremiumService",
                                    "Price": "300"
                                }
                            ],
                            "Constraints": {
                                "MultipleDimensionSelection": "Allowed",
                                "QuantityConfiguration": "Allowed"
                            }
                        }
                    ]
                }
            ]
        }
    ]
}
```

- Per i dettagli sull'API, consulta API Reference. StartChangeSetAWS SDK for Java 2.x

Aggiorna un'offerta per applicare una tariffa oraria annua

Il seguente esempio di codice mostra come aggiornare un'offerta per applicare prezzi orari annuali.

SDK per Java 2.x

> **Note**
>
> C'è altro da sapere. GitHub Trova l'esempio completo e scopri come configurarlo ed eseguirlo nel repository Marketplace AWS API Reference Code Library.

Per eseguire questo esempio, passa il seguente changeset JSON a **`RunChangesets`** in Utilities per avviare un changeset dalla sezione Utilities.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "offer-1111111111111"
            },
            "DetailsDocument": {
                "PricingModel": "Usage",
                "Terms": [
                    {
                        "Type": "UsageBasedPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "RateCard": [
                                    {
                                        "DimensionKey": "m5.large",
                                        "Price": "0.13"
                                    }
                                ]
                            }
                        ]
                    },
                    {
```

```
                        "Type": "ConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P365D"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "m5.large",
                                        "Price": "20.03"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        }
    ]
}
```

- Per i dettagli sull'API, consulta API Reference. StartChangeSetAWS SDK for Java 2.x

Aggiorna un'offerta per applicare il targeting a aree geografiche specifiche

Il seguente esempio di codice mostra come aggiornare un'offerta per applicare il targeting a aree geografiche specifiche.

SDK per Java 2.x

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri come configurarlo ed eseguirlo nel repository Marketplace AWS API Reference Code Library.

Per eseguire questo esempio, passa il seguente changeset JSON a **`RunChangesets`** in Utilities per avviare un changeset dalla sezione Utilities.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "offer-1111111111111"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "CountryCodes": [
                        "US",
                        "ES",
                        "FR",
                        "AU"
                    ]
                }
            }
        }
    ]
}
```

- Per i dettagli sull'API, consulta API Reference. StartChangeSetAWS SDK for Java 2.x

Aggiorna il nome e la descrizione di un'offerta pubblica

Il seguente esempio di codice mostra come aggiornare il nome e la descrizione di un'offerta pubblica.

SDK per Java 2.x

> **Note**
>
> C'è di più su GitHub. Trova l'esempio completo e scopri come configurarlo ed eseguirlo nel repository Marketplace AWS API Reference Code Library.

Per eseguire questo esempio, passa il seguente changeset JSON a **`RunChangesets`** in Utilities per avviare un changeset dalla sezione Utilities.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "offer-1111111111111"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        }
    ]
}
```

- Per i dettagli sull'API, consulta API Reference. StartChangeSetAWS SDK for Java 2.x

Aggiorna l'EULA di un'offerta

Il seguente esempio di codice mostra come aggiornare l'EULA di un'offerta.

SDK per Java 2.x

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri come configurarlo ed eseguirlo nel repository Marketplace AWS API Reference Code Library.

Per eseguire questo esempio, passa il seguente changeset JSON a **`RunChangesets`** in Utilities per avviare un changeset dalla sezione Utilities.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "offer-1111111111111"
            },
            "DetailsDocument": {
                "Name": "New offer name",
                "Description": "New offer description"
            }
        }
    ]
}
```

- Per i dettagli sull'API, consulta API Reference. StartChangeSetAWS SDK for Java 2.x

Aggiorna la data di scadenza di un'offerta privata a una data futura

Il seguente esempio di codice mostra come aggiornare la data di scadenza di un'offerta privata con una data futura per dare agli acquirenti più tempo per valutare e accettare l'offerta.

SDK per Java 2.x

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri come configurarlo ed eseguirlo nel repository Marketplace AWS API Reference Code Library.

Per eseguire questo esempio, passa il seguente changeset JSON a **`RunChangesets`** in Utilities per avviare un changeset dalla sezione Utilities.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "offer-1111111111111"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2026-01-01"
            }
        }
    ]
}
```

- Per i dettagli sull'API, consulta API Reference. StartChangeSetAWS SDK for Java 2.x

Aggiorna la durata della prova gratuita di un'offerta di prova gratuita pubblica per un prodotto SaaS

L'esempio di codice seguente mostra come aggiornare la durata della prova gratuita di un'offerta di prova gratuita pubblica per un prodotto SaaS.

SDK per Java 2.x

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri come configurarlo ed eseguirlo nel repository Marketplace AWS API Reference Code Library.

Per eseguire questo esempio, passa il seguente changeset JSON a **`RunChangesets`** in Utilities per avviare un changeset dalla sezione Utilities.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "offer-1111111111111"
            },
            "DetailsDocument": {
                "PricingModel": "Usage",
                "Terms": [
                    {
                        "Type": "FreeTrialPricingTerm",
                        "Duration": "P21D",
                        "Grants": [
                            {
                                "DimensionKey": "WorkloadSmall"
                            },
                            {
                                "DimensionKey": "WorkloadMedium"
                            }
                        ]
                    }
                ]
            }
        }
    ]
}
```

- Per i dettagli sull'API, consulta API Reference. StartChangeSetAWS SDK for Java 2.x

Aggiorna la politica di rimborso di un'offerta

Il seguente esempio di codice mostra come aggiornare la politica di rimborso di un'offerta.

SDK per Java 2.x

> **Note**
>
> C'è altro da sapere GitHub. Trova l'esempio completo e scopri come configurarlo ed eseguirlo nel repository Marketplace AWS API Reference Code Library.

Per eseguire questo esempio, passa il seguente changeset JSON a **`RunChangesets`** in Utilities per avviare un changeset dalla sezione Utilities.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "UpdateSupportTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "offer-1111111111111"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "SupportTerm",
                        "RefundPolicy": "Updated refund policy description"
                    }
                ]
            }
        }
    ]
}
```

- Per i dettagli sull'API, consulta API Reference. StartChangeSetAWS SDK for Java 2.x

Prodotti

Descrivi un prodotto AMI, SaaS o Container

Il seguente esempio di codice mostra come descrivere un prodotto AMI, SaaS o Container e verificare se contiene tutte le informazioni che desideri sapere sul prodotto.

SDK per Java 2.x

> **Note**
>
> C'è di più su. GitHub Trova l'esempio completo e scopri come configurarlo ed eseguirlo nel repository Marketplace AWS API Reference Code Library.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package com.example.awsmarketplace.catalogapi;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.*;
import com.example.awsmarketplace.utils.ReferenceCodesUtils;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.services.marketplacecatalog.MarketplaceCatalogClient;
import
 software.amazon.awssdk.services.marketplacecatalog.model.DescribeEntityRequest;
import
 software.amazon.awssdk.services.marketplacecatalog.model.DescribeEntityResponse;

public class DescribeEntity {

 /*
  * Describe my AMI or SaaS or Container product and check if it contains all the
 information I need to know about the product
  */
 public static void main(String[] args) {

  String offerId = args.length > 0 ? args[0] : OFFER_ID;

  DescribeEntityResponse describeEntityResponse =
 getDescribeEntityResponse(offerId);

  ReferenceCodesUtils.formatOutput(describeEntityResponse);
 }

 public static DescribeEntityResponse getDescribeEntityResponse(String offerId) {
  MarketplaceCatalogClient marketplaceCatalogClient =
    MarketplaceCatalogClient.builder()
```

```
    .httpClient(ApacheHttpClient.builder().build())
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

  DescribeEntityRequest describeEntityRequest =
    DescribeEntityRequest.builder()
    .catalog(AWS_MP_CATALOG)
    .entityId(offerId)
    .build();

  DescribeEntityResponse describeEntityResponse =
 marketplaceCatalogClient.describeEntity(describeEntityRequest);
  return describeEntityResponse;
 }
}
```

- Per i dettagli sull'API, consulta la sezione DescribeEntity AWS SDK for Java 2.xAPI Reference.

Elenca tutti i prodotti AMI, SaaS o Container e le offerte pubbliche associate

Il seguente esempio di codice mostra come elencare tutti i prodotti AMI, SaaS o Container e le offerte pubbliche associate.

SDK per Java 2.x

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri come configurarlo ed eseguirlo nel repository Marketplace AWS API Reference Code Library.

```
package com.example.awsmarketplace.catalogapi;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.*;
import com.example.awsmarketplace.utils.ReferenceCodesUtils;
```

```
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.services.marketplacecatalog.MarketplaceCatalogClient;
import software.amazon.awssdk.services.marketplacecatalog.model.EntitySummary;
import software.amazon.awssdk.services.marketplacecatalog.model.EntityTypeFilters;
import software.amazon.awssdk.services.marketplacecatalog.model.ListEntitiesRequest;
import
 software.amazon.awssdk.services.marketplacecatalog.model.ListEntitiesResponse;
import software.amazon.awssdk.services.marketplacecatalog.model.OfferFilters;
import
 software.amazon.awssdk.services.marketplacecatalog.model.OfferProductIdFilter;
import software.amazon.awssdk.services.marketplacecatalog.model.OfferStateFilter;
import
 software.amazon.awssdk.services.marketplacecatalog.model.OfferTargetingFilter;

public class ListEntities {

 /*
  * List all my AMI or SaaS or Container products and associated public offers
  */
 public static void main(String[] args) {

  Map<String, List<EntitySummary>> allProductsWithOffers =
 getAllProductsWithOffers();

  ReferenceCodesUtils.formatOutput(allProductsWithOffers);
 }

 public static Map<String, List<EntitySummary>> getAllProductsWithOffers() {
  MarketplaceCatalogClient marketplaceCatalogClient =
    MarketplaceCatalogClient.builder()
    .httpClient(ApacheHttpClient.builder().build())
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

  Map<String, List<EntitySummary>> allProductsWithOffers = new HashMap<String,
 List<EntitySummary>> ();

  // get all product entities
  List<EntitySummary> productEntityList = new ArrayList<EntitySummary>();

  ListEntitiesRequest listEntitiesRequest =
    ListEntitiesRequest.builder()
    .catalog(AWS_MP_CATALOG)
```

```
    .entityType(PRODUCT_TYPE_AMI)
    .maxResults(10)
    .nextToken(null)
    .build();


 ListEntitiesResponse listEntitiesResponse =
marketplaceCatalogClient.listEntities(listEntitiesRequest);

 productEntityList.addAll(listEntitiesResponse.entitySummaryList());


 while (listEntitiesResponse.nextToken() != null) {
  listEntitiesRequest =
    ListEntitiesRequest.builder()
    .catalog(AWS_MP_CATALOG)
    .entityType(PRODUCT_TYPE_AMI)
    .maxResults(10)
    .nextToken(listEntitiesResponse.nextToken())
    .build();
  listEntitiesResponse =
marketplaceCatalogClient.listEntities(listEntitiesRequest);
  productEntityList.addAll(listEntitiesResponse.entitySummaryList());
 }

 // loop through each product entity and get the public released offers associated
using product id filter

 for ( EntitySummary productEntitySummary : productEntityList) {
  EntityTypeFilters entityTypeFilters =
    EntityTypeFilters.builder()
    .offerFilters(OfferFilters.builder()
      .targeting(OfferTargetingFilter.builder()
        .valueListWithStrings(OFFER_TARGETING_NONE)
        .build())
      .state(OfferStateFilter.builder()
        .valueListWithStrings(OFFER_STATE_RELEASED)
        .build())
      .productId(OfferProductIdFilter.builder()
        .valueList(productEntitySummary.entityId())
        .build())
      .build())
    .build();
```

```
    listEntitiesRequest =
      ListEntitiesRequest.builder()
      .catalog(AWS_MP_CATALOG)
      .entityType(ENTITY_TYPE_OFFER)
      .maxResults(10)
      .entityTypeFilters(entityTypeFilters)
      .nextToken(null)
      .build();

    listEntitiesResponse =
  marketplaceCatalogClient.listEntities(listEntitiesRequest);

    // save all entitySummary of the results into entitySummaryList

    List<EntitySummary> offerEntitySummaryList = new ArrayList<EntitySummary>();

    offerEntitySummaryList.addAll(listEntitiesResponse.entitySummaryList());

    while ( listEntitiesResponse.nextToken() != null &&
  listEntitiesResponse.nextToken().length() > 0) {
     listEntitiesRequest =
       ListEntitiesRequest.builder()
       .catalog(AWS_MP_CATALOG)
       .entityType(ENTITY_TYPE_OFFER)
       .maxResults(10)
       .entityTypeFilters(entityTypeFilters)
       .nextToken(listEntitiesResponse.nextToken())
       .build();
     listEntitiesResponse =
  marketplaceCatalogClient.listEntities(listEntitiesRequest);
     offerEntitySummaryList.addAll(listEntitiesResponse.entitySummaryList());
    }

    // save final results into map; key = product id; value = offer entity summary
  list

    allProductsWithOffers.put(productEntitySummary.entityId(),
  offerEntitySummaryList);
   }
   return allProductsWithOffers;
  }

}
```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella Documentazione di riferimento delle API AWS SDK for Java 2.x .
  - DescribeEntity
  - ListEntities

Autorizzazione alla rivendita

Crea una bozza di autorizzazione alla rivendita

Il seguente esempio di codice mostra come creare una bozza di autorizzazione alla rivendita per qualsiasi tipo di prodotto in modo da poterla esaminare internamente prima della pubblicazione presso un Channel Partner.

SDK per Java 2.x

> **Note**
>
> C'è di più su. GitHub Trova l'esempio completo e scopri come configurarlo ed eseguirlo nel repository Marketplace AWS API Reference Code Library.

Per eseguire questo esempio, passa il seguente changeset JSON a **`RunChangesets`** in Utilities per avviare un changeset dalla sezione Utilities.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
```

```
                "ResellerAccountId": "111111111111"
            }
        }
    ]
}
```

- Per i dettagli sull'API, consulta API Reference. StartChangeSetAWS SDK for Java 2.x

Descrivi un'autorizzazione alla rivendita

Il seguente esempio di codice mostra come descrivere un'autorizzazione di rivendita.

SDK per Java 2.x

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri come configurarlo ed eseguirlo nel repository Marketplace AWS API Reference Code Library.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package com.example.awsmarketplace.catalogapi;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.*;
import com.example.awsmarketplace.utils.ReferenceCodesUtils;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.services.marketplacecatalog.MarketplaceCatalogClient;
import
 software.amazon.awssdk.services.marketplacecatalog.model.DescribeEntityRequest;
import
 software.amazon.awssdk.services.marketplacecatalog.model.DescribeEntityResponse;

public class DescribeEntity {

 /*
  * Describe my AMI or SaaS or Container product and check if it contains all the
 information I need to know about the product
  */
```

```
 public static void main(String[] args) {

  String offerId = args.length > 0 ? args[0] : OFFER_ID;

  DescribeEntityResponse describeEntityResponse =
 getDescribeEntityResponse(offerId);

  ReferenceCodesUtils.formatOutput(describeEntityResponse);
 }

 public static DescribeEntityResponse getDescribeEntityResponse(String offerId) {
  MarketplaceCatalogClient marketplaceCatalogClient =
    MarketplaceCatalogClient.builder()
    .httpClient(ApacheHttpClient.builder().build())
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

  DescribeEntityRequest describeEntityRequest =
    DescribeEntityRequest.builder()
    .catalog(AWS_MP_CATALOG)
    .entityId(offerId)
    .build();

  DescribeEntityResponse describeEntityResponse =
 marketplaceCatalogClient.describeEntity(describeEntityRequest);
  return describeEntityResponse;
 }
}
```

- Per i dettagli sull'API, consulta la sezione DescribeEntity AWS SDK for Java 2.xAPI Reference.

Pubblica un'autorizzazione di rivendita una tantum con un'offerta privata

Il seguente esempio di codice mostra come pubblicare un'autorizzazione di rivendita una tantum con un'offerta privata in modo che un Channel Partner possa utilizzarla per creare un'offerta privata di Channel Partner (CPPO).

SDK per Java 2.x

> **Note**
>
> C'è di più su. GitHub Trova l'esempio completo e scopri come configurarlo ed eseguirlo nel repository Marketplace AWS API Reference Code Library.

Per eseguire questo esempio, passa il seguente changeset JSON a **`RunChangesets`** in Utilities per avviare un changeset dalla sezione Utilities.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
            }
        },
        {
            "ChangeType": "ReleaseResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
```

```
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ResaleConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.small",
                                        "Price": "150"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
```

```
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "OffersMaxQuantity": 1
            }
        }
    ]
}
```

- Per i dettagli sull'API, consulta API Reference. StartChangeSetAWS SDK for Java 2.x

Pubblica un'autorizzazione di rivendita multiuso con una data di scadenza

Il seguente esempio di codice mostra come pubblicare un'autorizzazione di rivendita multiuso con una data di scadenza per un prodotto AMI con prezzi orari annuali in modo che un Channel Partner possa utilizzarla per creare un CPPO.

SDK per Java 2.x

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri come configurarlo ed eseguirlo nel repository Marketplace AWS API Reference Code Library.

Per eseguire questo esempio, passa il seguente changeset JSON a **`RunChangesets`** in Utilities per avviare un changeset dalla sezione Utilities.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateResaleAuthorization",
```

```
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ResaleConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
```

```
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.small",
                                        "Price": "150"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-05-31"
            }
        },
        {
            "ChangeType": "ReleaseResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

- Per i dettagli sull'API, consulta API Reference. StartChangeSetAWS SDK for Java 2.x

Pubblica un'autorizzazione di rivendita multiuso con una data di scadenza e un EULA

Il seguente esempio di codice mostra come pubblicare un'autorizzazione di rivendita multiuso con una data di scadenza per qualsiasi tipo di prodotto e aggiungere un EULA personalizzato da inviare all'acquirente.

SDK per Java 2.x

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri come configurarlo ed eseguirlo nel repository Marketplace AWS API Reference Code Library.

Per eseguire questo esempio, passa il seguente changeset JSON a **`RunChangesets`** in Utilities per avviare un changeset dalla sezione Utilities.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
            }
        },
        {
            "ChangeType": "ReleaseResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
```

```
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-05-31"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ResaleConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.small",
                                        "Price": "150"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
```

```
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        }
    ]
}
```

- Per i dettagli sull'API, consulta API Reference. StartChangeSetAWS SDK for Java 2.x

Pubblica l'autorizzazione alla rivendita multiuso con una data di scadenza e la documentazione del contratto di rivendita

Il seguente esempio di codice mostra come pubblicare un'autorizzazione di rivendita multiuso con una data di scadenza per qualsiasi tipo di prodotto e aggiungere la documentazione del contratto di rivendita tra l'ISV e il Channel Partner.

SDK per Java 2.x

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri come configurarlo ed eseguirlo nel repository Marketplace AWS API Reference Code Library.

Per eseguire questo esempio, passa il seguente changeset JSON a **`RunChangesets`** in Utilities per avviare un changeset dalla sezione Utilities.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
            }
        },
        {
            "ChangeType": "ReleaseResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-05-31"
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
```

```
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    },
                    {
                        "Type": "ResaleLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomResellerContract",
                                "Url": "https://s3.amazonaws.com/aws-mp-standard-
contracts/Standard-Contact-for-AWS-Marketplace-2022-07-14.pdf"}
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ResaleConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
```

```
                                    "DimensionKey": "t2.small",
                                    "Price": "150"
                                }
                            ],
                            "Constraints": {
                                "MultipleDimensionSelection": "Allowed",
                                "QuantityConfiguration": "Allowed"
                            }
                        }
                    ]
                }
            ]
        }
    ]
}
```

- Per i dettagli sull'API, consulta API Reference. StartChangeSetAWS SDK for Java 2.x

Pubblica un'autorizzazione di rivendita multiuso con scadenza e aggiungi un account acquirente specifico

Il seguente esempio di codice mostra come pubblicare un'autorizzazione di rivendita multiuso con una data di scadenza per qualsiasi tipo di prodotto e aggiungere un account acquirente specifico per la rivendita.

SDK per Java 2.x

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri come configurarlo ed eseguirlo nel repository Marketplace AWS API Reference Code Library.

Per eseguire questo esempio, passa il seguente changeset JSON a **`RunChangesets`** in Utilities per avviare un changeset dalla sezione Utilities.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
```

```
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
            }
        },
        {
            "ChangeType": "ReleaseResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-05-31"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ResaleConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
```

```
                            "Selector": {
                                "Type": "Duration",
                                "Value": "P12M"
                            },
                            "RateCard": [
                                {
                                    "DimensionKey": "t2.small",
                                    "Price": "150"
                                }
                            ],
                            "Constraints": {
                                "MultipleDimensionSelection": "Allowed",
                                "QuantityConfiguration": "Allowed"
                            }
                        }
                    ]
                }
            ]
        }
    },
    {
        "ChangeType": "UpdateBuyerTargetingTerms",
        "Entity": {
            "Type": "ResaleAuthorization@1.0",
            "Identifier": "$ResaleAuthorization.Entity.Identifier"
        },
        "DetailsDocument": {
            "Terms": [
                {
                    "Type": "BuyerTargetingTerm",
                    "PositiveTargeting": {
                        "BuyerAccounts": [
                            "111111111111"
                        ]
                    }
                }
            ]
        }
    },
    {
        "ChangeType": "UpdateLegalTerms",
        "Entity": {
            "Type": "ResaleAuthorization@1.0",
            "Identifier": "$ResaleAuthorization.Entity.Identifier"
```

```
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        }
    ]
}
```

- Per i dettagli sull'API, consulta API Reference. StartChangeSetAWS SDK for Java 2.x

Pubblica l'autorizzazione alla rivendita multiuso senza data di scadenza

Il seguente esempio di codice mostra come pubblicare un'autorizzazione di rivendita multiuso senza data di scadenza per un prodotto AMI con prezzi orari annuali in modo che un CP possa utilizzarla per creare un CPPO.

SDK per Java 2.x

> **Note**
>
> C'è di più su. GitHub Trova l'esempio completo e scopri come configurarlo ed eseguirlo nel repository Marketplace AWS API Reference Code Library.

Per eseguire questo esempio, passa il seguente changeset JSON a **`RunChangesets`** in Utilities per avviare un changeset dalla sezione Utilities.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
```

```
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
            }
        },
        {
            "ChangeType": "ReleaseResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ResaleConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.small",
                                        "Price": "150"
                                    }
```

```
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        }
    ]
}
```

- Per i dettagli sull'API, consulta API Reference. StartChangeSetAWS SDK for Java 2.x

Pubblica l'autorizzazione alla rivendita multiuso senza una data di scadenza e un EULA

Il seguente esempio di codice mostra come pubblicare un'autorizzazione di rivendita multiuso senza data di scadenza per qualsiasi tipo di prodotto e aggiungere un EULA personalizzato da inviare all'acquirente.

SDK per Java 2.x

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri come configurarlo ed eseguirlo nel repository Marketplace AWS API Reference Code Library.

Per eseguire questo esempio, passa il seguente changeset JSON a **`RunChangesets`** in Utilities per avviare un changeset dalla sezione Utilities.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
            }
        },
        {
            "ChangeType": "ReleaseResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
```

```
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ResaleConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.small",
                                        "Price": "150"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
```

```
                    }
                ]
            }
        }
    ]
}
```

- Per i dettagli sull'API, consulta API Reference. StartChangeSetAWS SDK for Java 2.x

Pubblica l'autorizzazione alla rivendita multiuso senza data di scadenza e la documentazione del contratto di rivendita

Il seguente esempio di codice mostra come pubblicare un'autorizzazione di rivendita multiuso senza data di scadenza per qualsiasi tipo di prodotto e aggiungere la documentazione del contratto di rivendita tra l'ISV e il Channel Partner.

SDK per Java 2.x

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri come configurarlo ed eseguirlo nel repository Marketplace AWS API Reference Code Library.

Per eseguire questo esempio, passa il seguente changeset JSON a **`RunChangesets`** in Utilities per avviare un changeset dalla sezione Utilities.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
```

```
            }
        },
        {
            "ChangeType": "ReleaseResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ResaleConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.small",
                                        "Price": "150"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
```

```
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    },
                    {
                        "Type": "ResaleLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomResellerContract",
                                "Url": "https://s3.amazonaws.com/aws-mp-standard-
contracts/Standard-Contact-for-AWS-Marketplace-2022-07-14.pdf"
                            }
                        ]
                    }
                ]
            }
        }
    ]
}
```

- Per i dettagli sull'API, consulta API Reference. StartChangeSetAWS SDK for Java 2.x

Pubblica l'autorizzazione di rivendita multiuso senza scadenza e aggiungi un account acquirente specifico

Il seguente esempio di codice mostra come pubblicare un'autorizzazione di rivendita multiuso senza data di scadenza per qualsiasi tipo di prodotto e aggiungere un account acquirente specifico per la rivendita.

SDK per Java 2.x

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri come configurarlo ed eseguirlo nel repository Marketplace AWS API Reference Code Library.

Per eseguire questo esempio, passa il seguente changeset JSON a **`RunChangesets`** in Utilities per avviare un changeset dalla sezione Utilities.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
            }
        },
        {
            "ChangeType": "ReleaseResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
```

```
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ResaleConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.small",
                                        "Price": "150"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateBuyerTargetingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerTargetingTerm",
                        "PositiveTargeting": {
                            "BuyerAccounts": [
                                "111111111111"
                            ]
                        }
                    }
                ]
```

```
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        }
    ]
}
```

- Per i dettagli sull'API, consulta API Reference. StartChangeSetAWS SDK for Java 2.x

Pubblica un'autorizzazione di rivendita una tantum e aggiungi un piano di pagamento flessibile

Il seguente esempio di codice mostra come pubblicare un'autorizzazione di rivendita una tantum per qualsiasi tipo di prodotto e aggiungere un piano di pagamento flessibile.

SDK per Java 2.x

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri come configurarlo ed eseguirlo nel repository Marketplace AWS API Reference Code Library.

Per eseguire questo esempio, passa il seguente changeset JSON a **`RunChangesets`** in Utilities per avviare un changeset dalla sezione Utilities.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
            }
        },
        {
            "ChangeType": "ReleaseResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ResaleFixedUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "Price": "0.00",
                        "Duration": "P12M",
                        "Grants": [
                          {
```

```
                        "DimensionKey": "Users",
                        "MaxQuantity": 10
                    }
                ]
            }
        ]
    }
},
{
    "ChangeType": "UpdatePaymentScheduleTerms",
    "Entity": {
        "Type": "ResaleAuthorization@1.0",
        "Identifier": "$ResaleAuthorization.Entity.Identifier"
    },
    "DetailsDocument": {
        "Terms": [
            {
                "Type": "ResalePaymentScheduleTerm",
                "CurrencyCode": "USD",
                "Schedule": [
                    {
                        "ChargeDate": "2023-09-01",
                        "ChargeAmount": "200.00"
                    },
                    {
                        "ChargeDate": "2023-12-01",
                        "ChargeAmount": "250.00"
                    }
                ]
            }
        ]
    }
},
{
    "ChangeType": "UpdateAvailability",
    "Entity": {
        "Type": "ResaleAuthorization@1.0",
        "Identifier": "$ResaleAuthorization.Entity.Identifier"
    },
    "DetailsDocument": {
        "AvailabilityEndDate": "2023-06-30",
        "OffersMaxQuantity": 1
    }
},
```

```
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        }
    ]
}
```

- Per i dettagli sull'API, consulta API Reference. StartChangeSetAWS SDK for Java 2.x

Pubblica un'autorizzazione di rivendita una tantum e aggiungi un EULA

Il seguente esempio di codice mostra come pubblicare un'autorizzazione di rivendita una tantum per qualsiasi tipo di prodotto e aggiungere un EULA personalizzato da inviare all'acquirente.

SDK per Java 2.x

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri come configurarlo ed eseguirlo nel repository Marketplace AWS API Reference Code Library.

Per eseguire questo esempio, passa il seguente changeset JSON a **`RunChangesets`** in Utilities per avviare un changeset dalla sezione Utilities.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
            }
        },
        {
            "ChangeType": "ReleaseResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "OffersMaxQuantity": 1
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
```

```
                    {
                        "Type": "ResaleConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.small",
                                        "Price": "150"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
```

```
            }
        }
    ]
}
```

- Per i dettagli sull'API, consulta API Reference. StartChangeSetAWS SDK for Java 2.x

Pubblica un'autorizzazione di rivendita una tantum e aggiungi un account acquirente specifico

Il seguente esempio di codice mostra come pubblicare un'autorizzazione di rivendita una tantum per qualsiasi tipo di prodotto e aggiungere un account acquirente specifico per la rivendita.

SDK per Java 2.x

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri come configurarlo ed eseguirlo nel repository Marketplace AWS API Reference Code Library.

Per eseguire questo esempio, passa il seguente changeset JSON a **`RunChangesets`** in Utilities per avviare un changeset dalla sezione Utilities.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
            }
        },
        {
            "ChangeType": "ReleaseResaleAuthorization",
            "Entity": {
```

```
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ResaleConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.small",
                                        "Price": "150"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
```

```
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "OffersMaxQuantity": "1"
            }
        },
        {
            "ChangeType": "UpdateBuyerTargetingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerTargetingTerm",
                        "PositiveTargeting": {
                            "BuyerAccounts": [
                                "111111111111"
                            ]
                        }
                    }
                ]
            }
```

```
        }
    ]
}
```

- Per i dettagli sull'API, consulta API Reference. StartChangeSetAWS SDK for Java 2.x

Pubblica l'autorizzazione alla rivendita una tantum e aggiungi la documentazione del contratto di rivendita

Il seguente esempio di codice mostra come pubblicare un'autorizzazione di rivendita una tantum per qualsiasi tipo di prodotto e aggiungere la documentazione del contratto di rivendita tra l'ISV e il Channel Partner.

SDK per Java 2.x

> **Note**
>
> C'è di più su. GitHub Trova l'esempio completo e scopri come configurarlo ed eseguirlo nel repository Marketplace AWS API Reference Code Library.

Per eseguire questo esempio, passa il seguente changeset JSON a **`RunChangesets`** in Utilities per avviare un changeset dalla sezione Utilities.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
            }
        },
        {
```

```
            "ChangeType": "ReleaseResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "OffersMaxQuantity": 1
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ResaleConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.small",
                                        "Price": "150"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
```

```
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        }
    ]
}
```

- Per i dettagli sull'API, consulta API Reference. StartChangeSetAWS SDK for Java 2.x

Pubblica un'autorizzazione di rivendita una tantum e aggiungi se si tratta di un rinnovo

Il seguente esempio di codice mostra come pubblicare un'autorizzazione di rivendita una tantum per qualsiasi tipo di prodotto e aggiungere se si tratta di un rinnovo.

SDK per Java 2.x

> **Note**
>
> C'è altro da sapere. GitHub Trova l'esempio completo e scopri come configurarlo ed eseguirlo nel repository Marketplace AWS API Reference Code Library.

Per eseguire questo esempio, passa il seguente changeset JSON a **`RunChangesets`** in Utilities per avviare un changeset dalla sezione Utilities.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
            }
        },
        {
            "ChangeType": "UpdateBuyerTargetingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerTargetingTerm",
                        "PositiveTargeting": {
                            "BuyerAccounts": [
                                "222222222222"
                            ]
                        }
```

```
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "OffersMaxQuantity": 1
            }
        },
        {
            "ChangeType":"UpdateInformation",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "PreExistingBuyerAgreement": {
                    "AcquisitionChannel": "AwsMarketplace",
                    "PricingModel": "Contract"
                }
             }
        }
    ]
}
```

- Per i dettagli sull'API, consulta API Reference. StartChangeSetAWS SDK for Java 2.x

Limita l'autorizzazione alla rivendita

Il seguente esempio di codice mostra come limitare l'autorizzazione alla rivendita.

SDK per Java 2.x

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri come configurarlo ed eseguirlo nel repository Marketplace AWS API Reference Code Library.

Per eseguire questo esempio, passa il seguente changeset JSON a **`RunChangesets`** in Utilities per avviare un changeset dalla sezione Utilities.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "RestrictResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "resaleauthz-1111111111111"
            },
            "DetailsDocument": {}
        }
    ]
}
```

- Per i dettagli sull'API, consulta API Reference. StartChangeSetAWS SDK for Java 2.x

Aggiorna il nome e la descrizione dell'autorizzazione di rivendita monouso o multiuso

Il seguente esempio di codice mostra come aggiornare il nome e la descrizione dell'autorizzazione di rivendita monouso o multiuso prima della pubblicazione per qualsiasi tipo di prodotto.

SDK per Java 2.x

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri come configurarlo ed eseguirlo nel repository Marketplace AWS API Reference Code Library.

Per eseguire questo esempio, passa il seguente changeset JSON a **`RunChangesets`** in Utilities per avviare un changeset dalla sezione Utilities.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType":"UpdateInformation",
            "Entity": {
             "Type": "ResaleAuthorization@1.0",
             "Identifier": "resaleauthz-1111111111111"
            },
            "DetailsDocument": {
             "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product"
            }
        }
    ]
}
```

- Per i dettagli sull'API, consulta API Reference. StartChangeSetAWS SDK for Java 2.x

Prodotti SaaS

Crea una bozza di prodotto SaaS con una bozza di offerta pubblica

Il seguente esempio di codice mostra come creare una bozza di prodotto SaaS con una bozza di offerta pubblica.

SDK per Java 2.x

> **Note**
>
> C'è di più su. GitHub Trova l'esempio completo e scopri come configurarlo ed eseguirlo nel repository Marketplace AWS API Reference Code Library.

Per eseguire questo esempio, passa il seguente changeset JSON a **`RunChangesets`** in Utilities per avviare un changeset dalla sezione Utilities.

```
{
```

```
    "Catalog":"AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateProduct",
            "ChangeName": "CreateProductChange",
            "Entity": {
                "Type": "SaaSProduct@1.0"
            },
            "DetailsDocument": {
              "ProductTitle": "Sample product"
            }
        },
        {
            "ChangeType": "CreateOffer",
            "ChangeName": "CreateOfferChange",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "DetailsDocument": {
                "ProductId": "$CreateProductChange.Entity.Identifier",
                "Name": "Test Offer"
            }
        }
    ]
}
```

- Per i dettagli sull'API, consulta API Reference. StartChangeSetAWS SDK for Java 2.x

Crea un prodotto SaaS pubblico o limitato e un'offerta pubblica con prezzi contrattuali

Il seguente esempio di codice mostra come creare un prodotto SaaS pubblico o limitato e un'offerta pubblica con prezzi contrattuali. Questo esempio crea un EULA standard o personalizzato.

SDK per Java 2.x

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri come configurarlo ed eseguirlo nel repository Marketplace AWS API Reference Code Library.

Per eseguire questo esempio, passa il seguente changeset JSON a **`RunChangesets`** in Utilities per avviare un changeset dalla sezione Utilities.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateProduct",
            "Entity": {
                "Type": "SaaSProduct@1.0"
            },
            "ChangeName": "CreateProductChange",
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "ProductTitle": "Sample product",
                "ShortDescription": "Brief description",
                "LongDescription": "Detailed description",
                "Highlights": [
                    "Sample highlight"
                ],
                "SearchKeywords": [
                    "Sample keyword"
                ],
                "Categories": [
                    "Data Catalogs"
                ],
                "LogoUrl": "https://s3.amazonaws.com/logos/sample.png",
                "VideoUrls": [
                    "https://sample.amazonaws.com/awsmp-video-1"
                ],
                "AdditionalResources": []
            }
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
```

```
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111",
                        "222222222222"
                    ]
                }
            }
        },
        {
            "ChangeType": "AddDeliveryOptions",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "DeliveryOptions": [
                    {
                        "Details": {
                            "SaaSUrlDeliveryOptionDetails": {
                                "FulfillmentUrl":"https://sample.amazonaws.com/
sample-saas-fulfillment-url"
                            }
                        }
                    }
                ]
            }
        },
        {
            "ChangeType": "AddDimensions",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": [
                {
                    "Key": "BasicService",
                    "Description": "Basic Service",
                    "Name": "Basic Service",
                    "Types": [
                        "Entitled"
```

```
                    ],
                    "Unit": "Units"
                },
                {
                    "Key": "PremiumService",
                    "Description": "Premium Service",
                    "Name": "Premium Service",
                    "Types": [
                        "Entitled"
                    ],
                    "Unit": "Units"
                }
            ]
        },
        {
            "ChangeType": "ReleaseProduct",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "CreateOffer",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "ChangeName": "CreateOfferChange",
            "DetailsDocument": {
                "ProductId": "$CreateProductChange.Entity.Identifier"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test public offer for SaaSProduct using AWS Marketplace API
Reference Code",
                "Description": "Test public offer with contract pricing for
SaaSProduct using AWS Marketplace API Reference Code"
            }
```

```
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P1M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "BasicService",
                                        "Price": "20"
                                    },
                                    {
                                        "DimensionKey": "PremiumService",
                                        "Price": "25"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            },
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "BasicService",
                                        "Price": "150"
                                    },
```

```
                                {
                                    "DimensionKey": "PremiumService",
                                    "Price": "300"
                                }
                            ],
                            "Constraints": {
                                "MultipleDimensionSelection": "Allowed",
                                "QuantityConfiguration": "Allowed"
                            }
                        }
                    ]
                }
            ]
        }
    },
    {
        "ChangeType": "UpdateLegalTerms",
        "Entity": {
            "Type": "Offer@1.0",
            "Identifier": "$CreateOfferChange.Entity.Identifier"
        },
        "DetailsDocument": {
            "Terms": [
                {
                    "Type": "LegalTerm",
                    "Documents": [
                        {
                            "Type": "StandardEula",
                            "Version": "2022-07-14"
                        }
                    ]
                }
            ]
        }
    },
    {
        "ChangeType": "UpdateSupportTerms",
        "Entity": {
            "Type": "Offer@1.0",
            "Identifier": "$CreateOfferChange.Entity.Identifier"
        },
        "DetailsDocument": {
            "Terms": [
                {
```

```
                        "Type": "SupportTerm",
                        "RefundPolicy": "Absolutely no refund, period."
                    }
                ]
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

- Per i dettagli sull'API, consulta API Reference. StartChangeSetAWS SDK for Java 2.x

Crea un prodotto SaaS pubblico o limitato e un'offerta pubblica con contratto con prezzi Pay-As-You-Go

Il seguente esempio di codice mostra come creare un prodotto SaaS pubblico o limitato e un'offerta pubblica con un contratto con Pay-As-You-Go prezzi. Questo esempio crea un EULA standard o personalizzato.

SDK per Java 2.x

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri come configurarlo ed eseguirlo nel repository Marketplace AWS API Reference Code Library.

Per eseguire questo esempio, passa il seguente changeset JSON a **`RunChangesets`** in Utilities per avviare un changeset dalla sezione Utilities.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
```

```
            "ChangeType": "CreateProduct",
            "Entity": {
                "Type": "SaaSProduct@1.0"
            },
            "ChangeName": "CreateProductChange",
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "ProductTitle": "Sample product",
                "ShortDescription": "Brief description",
                "LongDescription": "Detailed description",
                "Highlights": [
                    "Sample highlight"
                ],
                "SearchKeywords": [
                    "Sample keyword"
                ],
                "Categories": [
                    "Data Catalogs"
                ],
                "LogoUrl": "https://s3.amazonaws.com/logos/sample.png",
                "VideoUrls": [
                    "https://sample.amazonaws.com/awsmp-video-1"
                ],
                "AdditionalResources": []
            }
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111",
                        "222222222222"
```

```
                    ]
                }
            }
        },
        {
            "ChangeType": "AddDeliveryOptions",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "DeliveryOptions": [
                    {
                        "Details": {
                            "SaaSUrlDeliveryOptionDetails": {
                                "FulfillmentUrl":"https://sample.amazonaws.com/
sample-saas-fulfillment-url"
                            }
                        }
                    }
                ]
            }
        },
        {
            "ChangeType": "AddDimensions",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": [
                {
                    "Key": "BasicService",
                    "Description": "Basic Service",
                    "Name": "Basic Service",
                    "Types": [
                        "Entitled"
                    ],
                    "Unit": "Units"
                },
                {
                    "Key": "PremiumService",
                    "Description": "Premium Service",
                    "Name": "Premium Service",
                    "Types": [
```

```
                    "Entitled"
                ],
                "Unit": "Units"
            },
            {
                "Key": "WorkloadSmall",
                "Description": "Workload: Per medium instance",
                "Name": "Workload: Per medium instance",
                "Types": [
                    "ExternallyMetered"
                ],
                "Unit": "Units"
            },
            {
                "Key": "WorkloadMedium",
                "Description": "Workload: Per large instance",
                "Name": "Workload: Per large instance",
                "Types": [
                    "ExternallyMetered"
                ],
                "Unit": "Units"
            }
        ]
    },
    {
        "ChangeType": "ReleaseProduct",
        "Entity": {
            "Type": "SaaSProduct@1.0",
            "Identifier": "$CreateProductChange.Entity.Identifier"
        },
        "DetailsDocument": {}
    },
    {
        "ChangeType": "CreateOffer",
        "Entity": {
            "Type": "Offer@1.0"
        },
        "ChangeName": "CreateOfferChange",
        "DetailsDocument": {
            "ProductId": "$CreateProductChange.Entity.Identifier"
        }
    },
    {
        "ChangeType": "UpdateInformation",
```

```
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test public offer for SaaSProduct using AWS Marketplace API
Reference Code",
                "Description": "Test public offer with contract pricing for
SaaSProduct using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "UsageBasedPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "RateCard": [
                                    {
                                        "DimensionKey": "WorkloadSmall",
                                        "Price": "0.15"
                                    },
                                    {
                                        "DimensionKey": "WorkloadMedium",
                                        "Price": "0.25"
                                    }
                                ]
                            }
                        ]
                    },
                    {
                        "Type": "ConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
```

```
                                "Type": "Duration",
                                "Value": "P12M"
                            },
                            "RateCard": [
                                {
                                    "DimensionKey": "BasicService",
                                    "Price": "150"
                                },
                                {
                                    "DimensionKey": "PremiumService",
                                    "Price": "300"
                                }
                            ],
                            "Constraints": {
                                "MultipleDimensionSelection": "Allowed",
                                "QuantityConfiguration": "Allowed"
                            }
                        }
                    ]
                }
            ]
        }
    },
    {
        "ChangeType": "UpdateLegalTerms",
        "Entity": {
            "Type": "Offer@1.0",
            "Identifier": "$CreateOfferChange.Entity.Identifier"
        },
        "DetailsDocument": {
            "Terms": [
                {
                    "Type": "LegalTerm",
                    "Documents": [
                        {
                            "Type": "StandardEula",
                            "Version": "2022-07-14"
                        }
                    ]
                }
            ]
        }
    },
    {
```

```
            "ChangeType": "UpdateSupportTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "SupportTerm",
                        "RefundPolicy": "Absolutely no refund, period."
                    }
                ]
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

- Per i dettagli sull'API, consulta API Reference. StartChangeSetAWS SDK for Java 2.x

Crea un prodotto SaaS pubblico o limitato e un'offerta pubblica con prezzi di abbonamento

Il seguente esempio di codice mostra come creare un prodotto SaaS pubblico o limitato e un'offerta pubblica con prezzi di abbonamento. Questo esempio crea un EULA standard o personalizzato.

SDK per Java 2.x

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri come configurarlo ed eseguirlo nel repository Marketplace AWS API Reference Code Library.

Per eseguire questo esempio, passa il seguente changeset JSON a **`RunChangesets`** in Utilities per avviare un changeset dalla sezione Utilities.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateProduct",
            "Entity": {
                "Type": "SaaSProduct@1.0"
            },
            "ChangeName": "CreateProductChange",
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "ProductTitle": "Sample product",
                "ShortDescription": "Brief description",
                "LongDescription": "Detailed description",
                "Highlights": [
                    "Sample highlight"
                ],
                "SearchKeywords": [
                    "Sample keyword"
                ],
                "Categories": [
                    "Data Catalogs"
                ],
                "LogoUrl": "https://s3.amazonaws.com/logos/sample.png",
                "VideoUrls": [
                    "https://sample.amazonaws.com/awsmp-video-1"
                ],
                "AdditionalResources": []
            }
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
```

```
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111",
                        "222222222222"
                    ]
                }
            }
        },
        {
            "ChangeType": "AddDeliveryOptions",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "DeliveryOptions": [
                    {
                        "Details": {
                            "SaaSUrlDeliveryOptionDetails": {
                                "FulfillmentUrl":"https://sample.amazonaws.com/
sample-saas-fulfillment-url"
                            }
                        }
                    }
                ]
            }
        },
        {
            "ChangeType": "AddDimensions",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": [
                {
                    "Key": "WorkloadSmall",
                    "Description": "Workload: Per medium instance",
                    "Name": "Workload: Per medium instance",
                    "Types": [
                        "ExternallyMetered"
                    ],
                    "Unit": "Units"
                },
```

```
                {
                    "Key": "WorkloadMedium",
                    "Description": "Workload: Per large instance",
                    "Name": "Workload: Per large instance",
                    "Types": [
                        "ExternallyMetered"
                    ],
                    "Unit": "Units"
                }
            ]
        },
        {
            "ChangeType": "ReleaseProduct",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "CreateOffer",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "ChangeName": "CreateOfferChange",
            "DetailsDocument": {
                "ProductId": "$CreateProductChange.Entity.Identifier"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test public offer for SaaSProduct using AWS Marketplace API
Reference Code",
                "Description": "Test public offer with contract pricing for
SaaSProduct using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
```

```
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Usage",
                "Terms": [
                    {
                        "Type": "UsageBasedPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "RateCard": [
                                    {
                                        "DimensionKey": "WorkloadSmall",
                                        "Price": "0.15"
                                    },
                                    {
                                        "DimensionKey": "WorkloadMedium",
                                        "Price": "0.25"
                                    }
                                ]
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "StandardEula",
                                "Version": "2022-07-14"
                            }
                        ]
```

```
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateSupportTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "SupportTerm",
                        "RefundPolicy": "Absolutely no refund, period."
                    }
                ]
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

- Per i dettagli sull'API, consulta API Reference. StartChangeSetAWS SDK for Java 2.x

Pubblica un prodotto SaaS e un'offerta pubblica associata

Il seguente esempio di codice mostra come pubblicare un prodotto SaaS e un'offerta pubblica associata. Per impostazione predefinita, il prodotto sarà in uno stato limitato.

SDK per Java 2.x

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri come configurarlo ed eseguirlo nel repository Marketplace AWS API Reference Code Library.

Per eseguire questo esempio, passa il seguente changeset JSON a **`RunChangesets`** in Utilities per avviare un changeset dalla sezione Utilities.

```
{
    "Catalog":"AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateProduct",
            "ChangeName": "CreateProductChange",
            "Entity": {
                "Type": "SaaSProduct@1.0"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "ProductTitle": "Sample product",
                "ShortDescription": "Brief description",
                "LongDescription": "Detailed description",
                "Highlights": [
                    "Sample highlight"
                ],
                "SearchKeywords": [
                    "Sample keyword"
                ],
                "Categories": [
                    "Data Catalogs"
                ],
                "LogoUrl": "https://bucketname.s3.amazonaws.com/logo.png",
```

```
                "VideoUrls": [
                    "https://sample.amazonaws.com/awsmp-video-1"
                ],
                "AdditionalResources": []
            }
        },
        {
            "ChangeType": "AddDimensions",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": [
                {
                    "Key": "BasicService",
                    "Description": "Basic Service",
                    "Name": "Basic Service",
                    "Types": [
                        "Entitled"
                    ],
                    "Unit": "Units"
                },
                {
                    "Key": "PremiumService",
                    "Description": "Premium Service",
                    "Name": "Premium Service",
                    "Types": [
                        "Entitled"
                    ],
                    "Unit": "Units"
                }
            ]
        },
        {
            "ChangeType": "AddDeliveryOptions",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "DeliveryOptions": [
                    {
                        "Details": {
                            "SaaSUrlDeliveryOptionDetails": {
```

```
                                "FulfillmentUrl": "https://www.aws.amazon.com/
marketplace/management"
                            }
                        }
                    }
                ]
            }
        },
        {
            "ChangeType": "ReleaseProduct",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "CreateOffer",
            "ChangeName": "CreateOfferChange",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "DetailsDocument": {
                "ProductId": "$CreateProductChange.Entity.Identifier"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "New Test Offer",
                "Description": "New offer description"
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
```

```
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "StandardEula",
                                "Version": "2022-07-14"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateSupportTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "SupportTerm",
                        "RefundPolicy": "Updated refund policy description"
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
```

```
                                "Value": "P1M"
                            },
                            "RateCard": [
                                {
                                    "DimensionKey": "BasicService",
                                    "Price": "20"
                                },
                                {
                                    "DimensionKey": "PremiumService",
                                    "Price": "25"
                                }
                            ],
                            "Constraints": {
                                "MultipleDimensionSelection": "Allowed",
                                "QuantityConfiguration": "Allowed"
                            }
                        },
                        {
                            "Selector": {
                                "Type": "Duration",
                                "Value": "P12M"
                            },
                            "RateCard": [
                                {
                                    "DimensionKey": "BasicService",
                                    "Price": "150"
                                },
                                {
                                    "DimensionKey": "PremiumService",
                                    "Price": "300"
                                }
                            ],
                            "Constraints": {
                                "MultipleDimensionSelection": "Allowed",
                                "QuantityConfiguration": "Allowed"
                            }
                        }
                    ]
                }
            ]
        }
    },
    {
        "ChangeType": "UpdateRenewalTerms",
```

```
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "RenewalTerm"
                    }
                ]
            }
        },
        {
            "ChangeType":"ReleaseOffer",
            "Entity":{
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

- Per i dettagli sull'API, consulta API Reference. StartChangeSetAWS SDK for Java 2.x

Pubblica un prodotto SaaS e un'offerta pubblica associata da una bozza esistente

Il seguente esempio di codice mostra come pubblicare un prodotto SaaS e un'offerta pubblica associata da una bozza esistente. Per impostazione predefinita, il prodotto sarà in uno stato limitato.

SDK per Java 2.x

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri come configurarlo ed eseguirlo nel repository Marketplace AWS API Reference Code Library.

```
{
    "Catalog":"AWSMarketplace",
    "ChangeSet": [
```

```
        {
            "ChangeType": "UpdateVisibility",
            "ChangeName": "CreateProductChange",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "prod-1111111111111"
            },
            "DetailsDocument": {
                "TargetVisibility": "Public"
            }
        }
    ]
}
```

- Per i dettagli sull'API, consulta la sezione StartChangeSet AWS SDK for Java 2.xAPI Reference.

Aggiornamento delle dimensioni su un prodotto AMI o SaaS

Il seguente esempio di codice mostra come aggiornare le dimensioni su un prodotto AMI o SaaS.

SDK per Java 2.x

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri come configurarlo ed eseguirlo nel repository Marketplace AWS API Reference Code Library.

Per eseguire questo esempio, passa il seguente changeset JSON a **`RunChangesets`** in Utilities per avviare un changeset dalla sezione Utilities.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "UpdateDimensions",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "prod-111111111111"
            },
```

```
            "DetailsDocument": [
                {
                    "Key": "BasicService",
                    "Types": [
                        "Entitled"
                    ],
                    "Name": "Some new name",
                    "Description": "Some new description"
                }
            ]
        }
    ]
}
```

- Per i dettagli sull'API, consulta API Reference. StartChangeSetAWS SDK for Java 2.x

Utilità

Utilità per avviare un changeset

Il seguente esempio di codice mostra come definire le utilità per avviare un changeset.

SDK per Java 2.x

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri come configurarlo ed eseguirlo nel repository Marketplace AWS API Reference Code Library.

Utilità per caricare un changeset da un file JSON e iniziare a elaborarlo.

```
package com.example.awsmarketplace.catalogapi;

import java.io.ByteArrayInputStream;
import java.io.IOException;
import java.io.InputStream;
import java.util.ArrayList;
import java.util.List;

import org.apache.commons.io.IOUtils;
```

```
import org.apache.commons.lang3.StringUtils;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.core.document.Document;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.protocols.json.internal.unmarshall.document.DocumentUnmarshaller;
import software.amazon.awssdk.protocols.jsoncore.JsonNodeParser;
import software.amazon.awssdk.services.marketplacecatalog.MarketplaceCatalogClient;
import software.amazon.awssdk.services.marketplacecatalog.model.Change;
import software.amazon.awssdk.services.marketplacecatalog.model.Entity;
import
 software.amazon.awssdk.services.marketplacecatalog.model.StartChangeSetRequest;
import
 software.amazon.awssdk.services.marketplacecatalog.model.StartChangeSetResponse;
import com.google.gson.Gson;
import com.google.gson.GsonBuilder;
import com.google.gson.ToNumberPolicy;
import com.example.awsmarketplace.catalogapi.Entity.ChangeSet;
import com.example.awsmarketplace.catalogapi.Entity.ChangeSetEntity;
import com.example.awsmarketplace.catalogapi.Entity.Root;
import com.example.awsmarketplace.utils.ReferenceCodesUtils;
import com.example.awsmarketplace.utils.StringSerializer;

/**
 * Before running this Java V2 code example, convert all Details attribute to
 DetailsDocument if any
 */

public class RunChangesets {

 private static final Gson GSON = new GsonBuilder()
   .setObjectToNumberStrategy(ToNumberPolicy.LAZILY_PARSED_NUMBER)
   .registerTypeAdapter(String.class, new StringSerializer())
   .create();

 public static void main(String[] args) {

  // input json can be specified here or passed from input parameter
  String inputChangeSetFile = "changeSets/offers/
CreateReplacementOfferFromAGWithContractPricingDetailDocument.json";

  if (args.length > 0)
   inputChangeSetFile = args[0];
```

```
 // parse the input changeset file to string for process
 String changeSetsInput = readChangeSetToString(inputChangeSetFile);

 // process the changeset request
 try {
  StartChangeSetResponse result = getChangeSetRequestResult(changeSetsInput);
  ReferenceCodesUtils.formatOutput(result);
 } catch (Exception e) {
  e.printStackTrace();
 }
}

public static StartChangeSetResponse getChangeSetRequestResult(String
changeSetsInput) throws IOException {

 //set up AWS credentials
 MarketplaceCatalogClient marketplaceCatalogClient =
   MarketplaceCatalogClient.builder()
   .httpClient(ApacheHttpClient.builder().build())
   .credentialsProvider(ProfileCredentialsProvider.create())
   .build();

 //changeset list to save all the changesets in the changesets file
 List<Change> changeSetLists = new ArrayList<Change>();

 // read all changesets into object
 Root root = GSON.fromJson(changeSetsInput, Root.class);

 // process each changeset and add each changeset request to changesets list
 for (ChangeSet cs : root.changeSet) {

  ChangeSetEntity entity = cs.Entity;
  String entityType = entity.Type;
  String entityIdentifier = StringUtils.defaultIfBlank(entity.Identifier, null);
  Document detailsDocument = getDocumentFromObject(cs.DetailsDocument);

  Entity awsEntity =
    Entity.builder()
    .type(entityType)
    .identifier(entityIdentifier)
    .build();

  Change inputChangeRequest =
```

```
    Change.builder()
    .changeType(cs.ChangeType)
    .changeName(cs.ChangeName)
    .entity(awsEntity)
    .detailsDocument(detailsDocument)
    .build();

  changeSetLists.add(inputChangeRequest);
 }

 // process all changeset requests
 StartChangeSetRequest startChangeSetRequest =
   StartChangeSetRequest.builder()
   .catalog(root.catalog)
   .changeSet(changeSetLists)
   .build();

 StartChangeSetResponse result =
marketplaceCatalogClient.startChangeSet(startChangeSetRequest);

 return result;
}

public static Document getDocumentFromObject(Object detailsObject) {

 String detailsString = "{}";
 try {
  detailsString = IOUtils.toString(new
ByteArrayInputStream(GSON.toJson(detailsObject).getBytes()), "UTF-8");
 } catch (IOException e) {
  e.printStackTrace();
 }

 JsonNodeParser jsonNodeParser = JsonNodeParser.create();
 Document doc = jsonNodeParser.parse(detailsString).visit(new
DocumentUnmarshaller());
 return doc;
}


public static String readChangeSetToString (String inputChangeSetFile) {

 InputStream changesetInputStream =
RunChangesets.class.getClassLoader().getResourceAsStream(inputChangeSetFile);
```

```
  String changeSetsInput = null;

  try {
   changeSetsInput = IOUtils.toString(changesetInputStream, "UTF-8");
  } catch (IOException e) {
   e.printStackTrace();
  }

  return changeSetsInput;

 }
}
```

- Per i dettagli sull'API, consulta la sezione API StartChangeSetReference AWS SDK for Java 2.x .

Marketplace AWS Esempi di API Agreement che utilizzano 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 Marketplace AWS Agreement.

Ogni esempio include un collegamento al codice sorgente completo, in cui è possibile trovare istruzioni su come configurare ed eseguire il codice nel contesto.

Argomenti

- Accordi

Accordi

Ottieni un accordo completo IDs

Il seguente esempio di codice mostra come ottenere un accordo completo IDs.

SDK per Java 2.x

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri come configurarlo ed eseguirlo nel repository Marketplace AWS API Reference Code Library.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package com.example.awsmarketplace.agreementapi;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import
 software.amazon.awssdk.services.marketplaceagreement.model.AgreementViewSummary;
import software.amazon.awssdk.services.marketplaceagreement.model.Filter;
import
 software.amazon.awssdk.services.marketplaceagreement.model.SearchAgreementsRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.SearchAgreementsResponse;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.*;
import com.example.awsmarketplace.utils.ReferenceCodesUtils;

public class GetAllAgreementsIds {

 /*
  * Get all purchase agreements ids with party type = proposer;
  * Depend on the number of agreements in your account, this code may take some time
 to finish.
  */
 public static void main(String[] args) {

  List<String> agreementIds = getAllAgreementIds();
```

```
 ReferenceCodesUtils.formatOutput(agreementIds);

}

public static List<String> getAllAgreementIds() {
 MarketplaceAgreementClient marketplaceAgreementClient =
   MarketplaceAgreementClient.builder()
   .httpClient(ApacheHttpClient.builder().build())
   .credentialsProvider(ProfileCredentialsProvider.create())
   .build();

 // get all filters
 Filter partyType = Filter.builder().name(PARTY_TYPE_FILTER_NAME)
   .values(PARTY_TYPE_FILTER_VALUE_PROPOSER).build();

 Filter agreementType = Filter.builder().name(AGREEMENT_TYPE_FILTER_NAME)
   .values(AGREEMENT_TYPE_FILTER_VALUE_PURCHASEAGREEMENT).build();

 List<Filter> searchFilters = new ArrayList<Filter>();

 searchFilters.addAll(Arrays.asList(partyType, agreementType));

 // Save all results in a list array
 List<AgreementViewSummary> agreementSummaryList = new
ArrayList<AgreementViewSummary>();

 SearchAgreementsRequest searchAgreementsRequest =
   SearchAgreementsRequest.builder()
   .catalog(AWS_MP_CATALOG)
   .filters(searchFilters)
   .build();

 SearchAgreementsResponse searchAgreementsResponse =
marketplaceAgreementClient.searchAgreements(searchAgreementsRequest);

 agreementSummaryList.addAll(searchAgreementsResponse.agreementViewSummaries());

 while (searchAgreementsResponse.nextToken() != null &&
searchAgreementsResponse.nextToken().length() > 0) {
  searchAgreementsRequest =
    SearchAgreementsRequest.builder()
    .catalog(AWS_MP_CATALOG)
    .nextToken(searchAgreementsResponse.nextToken())
    .filters(searchFilters)
```

```
      .build();
    searchAgreementsResponse =
 marketplaceAgreementClient.searchAgreements(searchAgreementsRequest);
    agreementSummaryList.addAll(searchAgreementsResponse.agreementViewSummaries());
  }

  List<String> agreementIds = new ArrayList<String>();
  for (AgreementViewSummary summary : agreementSummaryList) {
   agreementIds.add(summary.agreementId());
  }
  return agreementIds;
 }

}
```

- Per i dettagli sull'API, consulta la sezione SearchAgreements AWS SDK for Java 2.xAPI Reference.

Ottieni tutti gli accordi

Il seguente esempio di codice mostra come ottenere tutti gli accordi.

SDK per Java 2.x

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri come configurarlo ed eseguirlo nel repository Marketplace AWS API Reference Code Library.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package com.example.awsmarketplace.agreementapi;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import
 software.amazon.awssdk.services.marketplaceagreement.model.AgreementViewSummary;
import software.amazon.awssdk.services.marketplaceagreement.model.Filter;
```

```
import
 software.amazon.awssdk.services.marketplaceagreement.model.SearchAgreementsRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.SearchAgreementsResponse;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.*;

import com.example.awsmarketplace.utils.ReferenceCodesUtils;

public class GetAllAgreements {

 /*
  * Get all purchase agreements with party type = proposer;
  * Depend on the number of agreements in your account, this code may take some time
 to finish.
  */
 public static void main(String[] args) {

  List<AgreementViewSummary> agreementSummaryList = getAllAgreements();

  ReferenceCodesUtils.formatOutput(agreementSummaryList);
 }

 public static List<AgreementViewSummary> getAllAgreements() {
  MarketplaceAgreementClient marketplaceAgreementClient =
    MarketplaceAgreementClient.builder()
    .httpClient(ApacheHttpClient.builder().build())
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

  // get all filters

  Filter partyType = Filter.builder().name(PARTY_TYPE_FILTER_NAME)
    .values(PARTY_TYPE_FILTER_VALUE_PROPOSER).build();

  Filter agreementType = Filter.builder().name(AGREEMENT_TYPE_FILTER_NAME)
    .values(AGREEMENT_TYPE_FILTER_VALUE_PURCHASEAGREEMENT).build();

  List<Filter> searchFilters = new ArrayList<Filter>();
```

```
  searchFilters.addAll(Arrays.asList(partyType, agreementType));

  // Save all results in a list array

  List<AgreementViewSummary> agreementSummaryList = new
 ArrayList<AgreementViewSummary>();

  SearchAgreementsRequest searchAgreementsRequest =
    SearchAgreementsRequest.builder()
    .catalog(AWS_MP_CATALOG)
    .filters(searchFilters)
    .build();

  SearchAgreementsResponse searchAgreementsResponse =
 marketplaceAgreementClient.searchAgreements(searchAgreementsRequest);

  agreementSummaryList.addAll(searchAgreementsResponse.agreementViewSummaries());

  while (searchAgreementsResponse.nextToken() != null &&
 searchAgreementsResponse.nextToken().length() > 0) {
   searchAgreementsRequest =
     SearchAgreementsRequest.builder()
     .catalog(AWS_MP_CATALOG)
     .nextToken(searchAgreementsResponse.nextToken())
     .filters(searchFilters).build();
   searchAgreementsResponse =
 marketplaceAgreementClient.searchAgreements(searchAgreementsRequest);
   agreementSummaryList.addAll(searchAgreementsResponse.agreementViewSummaries());
  }
  return agreementSummaryList;
 }

}
```

- Per i dettagli sull'API, consulta la sezione SearchAgreements AWS SDK for Java 2.xAPI Reference.

Ottieni l'ID cliente da un contratto

Il seguente esempio di codice mostra come ottenere l'ID cliente da un contratto.

SDK per Java 2.x

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri come configurarlo ed eseguirlo nel repository Marketplace AWS API Reference Code Library.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package com.example.awsmarketplace.agreementapi;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.*;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import
 software.amazon.awssdk.services.marketplaceagreement.model.DescribeAgreementRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.DescribeAgreementResponse;

public class GetAgreementCustomerInfo {

 /*
  * Obtain metadata about the customer who created the agreement, such as the
 customer's AWS Account ID
  */
 public static void main(String[] args) {

  String agreementId = args.length > 0 ? args[0] : AGREEMENT_ID;

  DescribeAgreementResponse describeAgreementResponse =
 getDescribeAgreementResponse(agreementId);

  System.out.println("Customer's AWS Account ID is " +
 describeAgreementResponse.acceptor().accountId());

 }
```

```
 public static DescribeAgreementResponse getDescribeAgreementResponse(String
 agreementId) {
  MarketplaceAgreementClient marketplaceAgreementClient =
    MarketplaceAgreementClient.builder()
    .httpClient(ApacheHttpClient.builder().build())
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

  DescribeAgreementRequest describeAgreementRequest =
    DescribeAgreementRequest.builder()
    .agreementId(agreementId)
    .build();

  DescribeAgreementResponse describeAgreementResponse =
 marketplaceAgreementClient.describeAgreement(describeAgreementRequest);
  return describeAgreementResponse;
 }

}
```

- Per i dettagli sull'API, consulta la sezione DescribeAgreement AWS SDK for Java 2.xAPI Reference.

Ottieni i dettagli finanziari da un contratto

Il seguente esempio di codice mostra come ottenere dettagli finanziari da un accordo.

SDK per Java 2.x

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri come configurarlo ed eseguirlo nel repository Marketplace AWS API Reference Code Library.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package com.example.awsmarketplace.agreementapi;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.*;
```

```
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import
 software.amazon.awssdk.services.marketplaceagreement.model.DescribeAgreementRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.DescribeAgreementResponse;

public class GetAgreementFinancialDetails {

 /*
  * Obtain financial details, such as Total Contract Value of the agreement from a
 given agreement
  */
 public static void main(String[] args) {

  String agreementId = args.length > 0 ? args[0] : AGREEMENT_ID;

  String totalContractValue = getTotalContractValue(agreementId);

  System.out.println("Total Contract Value is " + totalContractValue);

 }

 public static String getTotalContractValue(String agreementId) {
  MarketplaceAgreementClient marketplaceAgreementClient =
    MarketplaceAgreementClient.builder()
    .httpClient(ApacheHttpClient.builder().build())
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

  DescribeAgreementRequest describeAgreementRequest =
    DescribeAgreementRequest.builder()
    .agreementId(agreementId)
    .build();

  DescribeAgreementResponse describeAgreementResponse =
 marketplaceAgreementClient.describeAgreement(describeAgreementRequest);

  String totalContractValue = "N/A";

  if ( describeAgreementResponse.estimatedCharges() != null ) {
```

```
    totalContractValue =
 describeAgreementResponse.estimatedCharges().agreementValue()
     + " "
     + describeAgreementResponse.estimatedCharges().currencyCode();
  }
  return totalContractValue;
 }
}
```

- Per i dettagli sull'API, consulta la sezione DescribeAgreement AWS SDK for Java 2.xAPI Reference.

Ottieni i dettagli della prova gratuita tramite un contratto

Il seguente esempio di codice mostra come ottenere i dettagli della prova gratuita da un contratto.

SDK per Java 2.x

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri come configurarlo ed eseguirlo nel repository Marketplace AWS API Reference Code Library.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package com.example.awsmarketplace.agreementapi;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import software.amazon.awssdk.services.marketplaceagreement.model.AcceptedTerm;
import
 software.amazon.awssdk.services.marketplaceagreement.model.FreeTrialPricingTerm;
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementTermsRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementTermsResponse;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.AGREEMENT_ID;
```

```
import java.util.ArrayList;
import java.util.List;

import com.example.awsmarketplace.utils.ReferenceCodesUtils;

public class GetAgreementTermsFreeTrialDetails {

 /*
  * Obtain the details from an agreement of a free trial I have provided to the
 customer
  */
 public static void main(String[] args) {

  String agreementId = args.length > 0 ? args[0] : AGREEMENT_ID;

  List<FreeTrialPricingTerm> freeTrialPricingTerms =
 getFreeTrialPricingTerms(agreementId);

  ReferenceCodesUtils.formatOutput(freeTrialPricingTerms);
 }

 public static List<FreeTrialPricingTerm> getFreeTrialPricingTerms(String
 agreementId) {
  MarketplaceAgreementClient marketplaceAgreementClient =
    MarketplaceAgreementClient.builder()
    .httpClient(ApacheHttpClient.builder().build())
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

  GetAgreementTermsRequest getAgreementTermsRequest =
    GetAgreementTermsRequest.builder().agreementId(agreementId)
     .build();

  GetAgreementTermsResponse getAgreementTermsResponse =
 marketplaceAgreementClient.getAgreementTerms(getAgreementTermsRequest);

  List<FreeTrialPricingTerm> freeTrialPricingTerms = new
 ArrayList<FreeTrialPricingTerm>();

  for (AcceptedTerm acceptedTerm : getAgreementTermsResponse.acceptedTerms()) {
   if (acceptedTerm.freeTrialPricingTerm() != null) {
    freeTrialPricingTerms.add(acceptedTerm.freeTrialPricingTerm());
   }
```

```
  }
  return freeTrialPricingTerms;
 }
}
```

- Per i dettagli sull'API, consulta la sezione DescribeAgreement AWS SDK for Java 2.xAPI Reference.

Ottieni informazioni su un contratto

Il seguente esempio di codice mostra come ottenere informazioni su un accordo.

SDK per Java 2.x

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri come configurarlo ed eseguirlo nel repository Marketplace AWS API Reference Code Library.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package com.example.awsmarketplace.agreementapi;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.*;
import com.example.awsmarketplace.utils.ReferenceCodesUtils;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import
 software.amazon.awssdk.services.marketplaceagreement.model.DescribeAgreementRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.DescribeAgreementResponse;

public class DescribeAgreement {

 public static void main(String[] args) {

  String agreementId = args.length > 0 ? args[0] : AGREEMENT_ID;
```

```
 DescribeAgreementResponse describeAgreementResponse = getResponse(agreementId);

 ReferenceCodesUtils.formatOutput(describeAgreementResponse);

}

public static DescribeAgreementResponse getResponse(String agreementId) {
 MarketplaceAgreementClient marketplaceAgreementClient =
   MarketplaceAgreementClient.builder()
   .httpClient(ApacheHttpClient.builder().build())
   .credentialsProvider(ProfileCredentialsProvider.create())
   .build();

 DescribeAgreementRequest describeAgreementRequest =
   DescribeAgreementRequest.builder()
   .agreementId(agreementId)
   .build();

 DescribeAgreementResponse describeAgreementResponse =
marketplaceAgreementClient.describeAgreement(describeAgreementRequest);
 return describeAgreementResponse;
}

}
```

- Per i dettagli sull'API, consulta la sezione DescribeAgreement AWS SDK for Java 2.xAPI Reference.

Ottieni i dettagli del prodotto e dell'offerta da un contratto

Il seguente esempio di codice mostra come ottenere i dettagli del prodotto e dell'offerta da un contratto.

SDK per Java 2.x

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri come configurarlo ed eseguirlo nel repository Marketplace AWS API Reference Code Library.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package com.example.awsmarketplace.agreementapi;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import
 software.amazon.awssdk.services.marketplaceagreement.model.DescribeAgreementRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.DescribeAgreementResponse;
import software.amazon.awssdk.services.marketplaceagreement.model.Resource;

import java.util.ArrayList;
import java.util.List;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.*;
import com.example.awsmarketplace.utils.ReferenceCodesUtils;

import software.amazon.awssdk.services.marketplacecatalog.MarketplaceCatalogClient;
import
 software.amazon.awssdk.services.marketplacecatalog.model.DescribeEntityRequest;
import
 software.amazon.awssdk.services.marketplacecatalog.model.DescribeEntityResponse;

public class GetProductAndOfferDetailFromAgreement {

 public static void main(String[] args) {

  // call Agreement API to get offer and product information for the agreement

  String agreementId = args.length > 0 ? args[0] : AGREEMENT_ID;

  List<DescribeEntityResponse> entityResponseList = getEntities(agreementId);

  for (DescribeEntityResponse response : entityResponseList) {
   ReferenceCodesUtils.formatOutput(response);
  }
 }

 public static List<DescribeEntityResponse> getEntities(String agreementId) {
```

```
 List<DescribeEntityResponse> entityResponseList = new
ArrayList<DescribeEntityResponse> ();

 MarketplaceAgreementClient marketplaceAgreementClient =
   MarketplaceAgreementClient.builder()
   .httpClient(ApacheHttpClient.builder().build())
   .credentialsProvider(ProfileCredentialsProvider.create())
   .build();

 DescribeAgreementRequest describeAgreementRequest =
   DescribeAgreementRequest.builder()
   .agreementId(agreementId)
   .build();

 DescribeAgreementResponse describeAgreementResponse =
marketplaceAgreementClient.describeAgreement(describeAgreementRequest);

 // get offer id for the given agreement

 String offerId = describeAgreementResponse.proposalSummary().offerId();

 // get all the product ids for this agreement

 List<String> productIds = new ArrayList<String>();
 for (Resource resource : describeAgreementResponse.proposalSummary().resources())
{
  productIds.add(resource.id());
 }

 // call Catalog API to get the details of the offer and products

 MarketplaceCatalogClient marketplaceCatalogClient =
   MarketplaceCatalogClient.builder()
   .httpClient(ApacheHttpClient.builder().build())
   .credentialsProvider(ProfileCredentialsProvider.create())
   .build();

 DescribeEntityRequest describeEntityRequest =
   DescribeEntityRequest.builder()
   .catalog(AWS_MP_CATALOG)
   .entityId(offerId).build();

 DescribeEntityResponse describeEntityResponse =
marketplaceCatalogClient.describeEntity(describeEntityRequest);
```

```
  entityResponseList.add(describeEntityResponse);

  for (String productId : productIds) {
   describeEntityRequest =
     DescribeEntityRequest.builder()
     .catalog(AWS_MP_CATALOG)
     .entityId(productId).build();
   describeEntityResponse =
 marketplaceCatalogClient.describeEntity(describeEntityRequest);
   System.out.println("Print details for product " + productId);
   entityResponseList.add(describeEntityResponse);
  }
  return entityResponseList;
 }
}
```

- Per i dettagli sull'API, consulta la sezione DescribeAgreement AWS SDK for Java 2.xAPI Reference.

Ottieni l'EULA di un contratto

Il seguente esempio di codice mostra come ottenere l'EULA di un accordo.

SDK per Java 2.x

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri come configurarlo ed eseguirlo nel repository Marketplace AWS API Reference Code Library.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package com.example.awsmarketplace.agreementapi;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import software.amazon.awssdk.services.marketplaceagreement.model.DocumentItem;
```

```
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementTermsRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementTermsResponse;

import java.util.ArrayList;
import java.util.List;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.AGREEMENT_ID;
import com.example.awsmarketplace.utils.ReferenceCodesUtils;

public class GetAgreementTermsEula {

 /*
  * Obtain the EULA I have entered into with my customer via the agreement
  */
 public static void main(String[] args) {

  String agreementId = args.length > 0 ? args[0] : AGREEMENT_ID;

  List<DocumentItem> legalEulaArray = getLegalEula(agreementId);

  ReferenceCodesUtils.formatOutput(legalEulaArray);
 }

 public static List<DocumentItem> getLegalEula(String agreementId) {
  MarketplaceAgreementClient marketplaceAgreementClient =
    MarketplaceAgreementClient.builder()
    .httpClient(ApacheHttpClient.builder().build())
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

  GetAgreementTermsRequest getAgreementTermsRequest =
    GetAgreementTermsRequest.builder().agreementId(agreementId)
    .build();

  GetAgreementTermsResponse getAgreementTermsResponse =
 marketplaceAgreementClient.getAgreementTerms(getAgreementTermsRequest);

  List<DocumentItem> legalEulaArray = new ArrayList<>();

  getAgreementTermsResponse.acceptedTerms().stream()
      .filter(acceptedTerm -> acceptedTerm.legalTerm() != null &&
 acceptedTerm.legalTerm().hasDocuments())
```

```
        .flatMap(acceptedTerm -> acceptedTerm.legalTerm().documents().stream())
        .filter(docItem -> docItem.type() != null)
        .forEach(legalEulaArray::add);
    return legalEulaArray;
  }

}
```

- Per i dettagli sull'API, consulta la sezione GetAgreementTerms AWS SDK for Java 2.xAPI Reference.

Ottieni i termini di rinnovo automatico di un contratto

Il seguente esempio di codice mostra come ottenere i termini di rinnovo automatico di un contratto.

SDK per Java 2.x

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri come configurarlo ed eseguirlo nel repository Marketplace AWS API Reference Code Library.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package com.example.awsmarketplace.agreementapi;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.*;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import software.amazon.awssdk.services.marketplaceagreement.model.AcceptedTerm;
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementTermsRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementTermsResponse;

public class GetAgreementAutoRenewal {
```

```
 /*
  * Obtain the auto-renewal status of the agreement
  */

 public static void main(String[] args) {

  String agreementId = args.length > 0 ? args[0] : AGREEMENT_ID;

  String autoRenewal = getAutoRenewal(agreementId);

  System.out.println("Auto-Renewal status is " + autoRenewal);
 }

 public static String getAutoRenewal(String agreementId) {
  MarketplaceAgreementClient marketplaceAgreementClient =
    MarketplaceAgreementClient.builder()
    .httpClient(ApacheHttpClient.builder().build())
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

  GetAgreementTermsRequest getAgreementTermsRequest =
    GetAgreementTermsRequest.builder()
    .agreementId(agreementId)
    .build();

  GetAgreementTermsResponse getAgreementTermsResponse =
 marketplaceAgreementClient.getAgreementTerms(getAgreementTermsRequest);

  String autoRenewal = "No Auto Renewal";

  for (AcceptedTerm acceptedTerm : getAgreementTermsResponse.acceptedTerms()) {
   if (acceptedTerm.renewalTerm() != null &&
 acceptedTerm.renewalTerm().configuration() != null
     && acceptedTerm.renewalTerm().configuration().enableAutoRenew() != null) {
    autoRenewal =
 String.valueOf(acceptedTerm.renewalTerm().configuration().enableAutoRenew().booleanValue())
    break;
   }
  }
  return autoRenewal;
 }

}
```

- Per i dettagli sull'API, consulta la sezione GetAgreementTerms AWS SDK for Java 2.xAPI Reference.

Ottieni le dimensioni acquistate in base a un contratto

Il seguente esempio di codice mostra come ottenere le dimensioni acquistate in un contratto.

SDK per Java 2.x

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri come configurarlo ed eseguirlo nel repository Marketplace AWS API Reference Code Library.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package com.example.awsmarketplace.agreementapi;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import software.amazon.awssdk.services.marketplaceagreement.model.AcceptedTerm;
import software.amazon.awssdk.services.marketplaceagreement.model.Dimension;
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementTermsRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementTermsResponse;

import java.util.ArrayList;
import java.util.List;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.AGREEMENT_ID;
import com.example.awsmarketplace.utils.ReferenceCodesUtils;

public class GetAgreementTermsDimensionPurchased {

 /*
  * Obtain the dimensions the buyer has purchased from me via the agreement
```

```
 */
public static void main(String[] args) {

 String agreementId = args.length > 0 ? args[0] : AGREEMENT_ID;

 List<String> dimensionKeys = getDimensionKeys(agreementId);

 ReferenceCodesUtils.formatOutput(dimensionKeys);
}

public static List<String> getDimensionKeys(String agreementId) {
 MarketplaceAgreementClient marketplaceAgreementClient =
   MarketplaceAgreementClient.builder()
   .httpClient(ApacheHttpClient.builder().build())
   .credentialsProvider(ProfileCredentialsProvider.create())
   .build();

 GetAgreementTermsRequest getAgreementTermsRequest =
   GetAgreementTermsRequest.builder().agreementId(agreementId)
   .build();

 GetAgreementTermsResponse getAgreementTermsResponse =
marketplaceAgreementClient.getAgreementTerms(getAgreementTermsRequest);

 List<String> dimensionKeys = new ArrayList<String>();
 for (AcceptedTerm acceptedTerm : getAgreementTermsResponse.acceptedTerms()) {
  if (acceptedTerm.configurableUpfrontPricingTerm() != null) {
   if
(acceptedTerm.configurableUpfrontPricingTerm().configuration().selectorValue() !=
null) {
    List<Dimension> dimensions =
acceptedTerm.configurableUpfrontPricingTerm().configuration().dimensions();
    for (Dimension dimension : dimensions) {
     dimensionKeys.add(dimension.dimensionKey());
    }
   }

  }
 }
 return dimensionKeys;
}
}
```

- Per i dettagli sull'API, consulta la sezione GetAgreementTerms AWS SDK for Java 2.xAPI Reference.

Ottieni le istanze di ogni dimensione acquistate in un contratto

Il seguente esempio di codice mostra come ottenere le istanze di ogni dimensione acquistate in un contratto.

SDK per Java 2.x

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri come configurarlo ed eseguirlo nel repository Marketplace AWS API Reference Code Library.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package com.example.awsmarketplace.agreementapi;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import software.amazon.awssdk.services.marketplaceagreement.model.AcceptedTerm;
import software.amazon.awssdk.services.marketplaceagreement.model.Dimension;
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementTermsRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementTermsResponse;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.AGREEMENT_ID;
import com.example.awsmarketplace.utils.ReferenceCodesUtils;

public class GetAgreementTermsDimensionInstances {
```

```
/*
 * get instances of each dimension that buyer has purchased in the agreement
 */
public static void main(String[] args) {

 String agreementId = args.length > 0 ? args[0] : AGREEMENT_ID;

 Map<String, List<Dimension>> dimensionMap = getDimensions(agreementId);

 ReferenceCodesUtils.formatOutput(dimensionMap);
}

public static Map<String, List<Dimension>> getDimensions(String agreementId) {
 MarketplaceAgreementClient marketplaceAgreementClient =
   MarketplaceAgreementClient.builder()
   .httpClient(ApacheHttpClient.builder().build())
   .credentialsProvider(ProfileCredentialsProvider.create())
   .build();

 GetAgreementTermsRequest getAgreementTermsRequest =
   GetAgreementTermsRequest.builder().agreementId(agreementId)
   .build();

 GetAgreementTermsResponse getAgreementTermsResponse =
marketplaceAgreementClient.getAgreementTerms(getAgreementTermsRequest);

 Map<String, List<Dimension>> dimensionMap = new HashMap<String,
List<Dimension>>();

 for (AcceptedTerm acceptedTerm : getAgreementTermsResponse.acceptedTerms()) {
  List<Dimension> dimensionsList = new ArrayList<Dimension>();
  if (acceptedTerm.configurableUpfrontPricingTerm() != null) {
   String selectorValue = "";
   if (acceptedTerm.configurableUpfrontPricingTerm().configuration() != null) {
    if
(acceptedTerm.configurableUpfrontPricingTerm().configuration().selectorValue() !=
null) {
     selectorValue =
acceptedTerm.configurableUpfrontPricingTerm().configuration().selectorValue();
    }
    if
(acceptedTerm.configurableUpfrontPricingTerm().configuration().hasDimensions()) {
     dimensionsList =
acceptedTerm.configurableUpfrontPricingTerm().configuration().dimensions();
```

```
     }
    }
    if (selectorValue.length() > 0) {
     dimensionMap.put(selectorValue, dimensionsList);
    }
   }
  }
  return dimensionMap;
 }
}
```

- Per i dettagli sull'API, consulta la sezione GetAgreementTerms AWS SDK for Java 2.xAPI Reference.

Ottieni il piano di pagamento di un contratto

Il seguente esempio di codice mostra come ottenere la tempistica di pagamento di un contratto.

SDK per Java 2.x

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri come configurarlo ed eseguirlo nel repository Marketplace AWS API Reference Code Library.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package com.example.awsmarketplace.agreementapi;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import software.amazon.awssdk.services.marketplaceagreement.model.AcceptedTerm;
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementTermsRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementTermsResponse;
import
 software.amazon.awssdk.services.marketplaceagreement.model.PaymentScheduleTerm;
```

```
import software.amazon.awssdk.services.marketplaceagreement.model.ScheduleItem;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.*;
import com.example.awsmarketplace.utils.ReferenceCodesUtils;

public class GetAgreementTermsPaymentSchedule {

 /*
  * Obtain the payment schedule I have agreed to with the agreement, including the
 invoice date and invoice amount
  */
 public static void main(String[] args) {

  String agreementId = args.length > 0 ? args[0] : AGREEMENT_ID;

  List<Map<String, Object>> paymentScheduleArray = getPaymentSchedules(agreementId);

  ReferenceCodesUtils.formatOutput(paymentScheduleArray);
 }

 public static List<Map<String, Object>> getPaymentSchedules(String agreementId) {
  MarketplaceAgreementClient marketplaceAgreementClient =
    MarketplaceAgreementClient.builder()
    .httpClient(ApacheHttpClient.builder().build())
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

  GetAgreementTermsRequest getAgreementTermsRequest =
    GetAgreementTermsRequest.builder().agreementId(agreementId)
    .build();

  GetAgreementTermsResponse getAgreementTermsResponse =
 marketplaceAgreementClient.getAgreementTerms(getAgreementTermsRequest);
  List<Map<String, Object>> paymentScheduleArray = new ArrayList<>();

  String currencyCode = "";

  for (AcceptedTerm acceptedTerm : getAgreementTermsResponse.acceptedTerms()) {
   if (acceptedTerm.paymentScheduleTerm() != null) {
```

```
    PaymentScheduleTerm paymentScheduleTerm = acceptedTerm.paymentScheduleTerm();
    if (paymentScheduleTerm.currencyCode() != null) {
     currencyCode = paymentScheduleTerm.currencyCode();
    }
    if (paymentScheduleTerm.hasSchedule()) {
     for (ScheduleItem schedule : paymentScheduleTerm.schedule()) {
      if (schedule.chargeDate() != null) {
       String chargeDate = schedule.chargeDate().toString();
       String chargeAmount = schedule.chargeAmount();
       Map<String, Object> scheduleMap = new HashMap<>();
       scheduleMap.put(ATTRIBUTE_CURRENCY_CODE, currencyCode);
       scheduleMap.put(ATTRIBUTE_CHARGE_DATE, chargeDate);
       scheduleMap.put(ATTRIBUTE_CHARGE_AMOUNT, chargeAmount);
       paymentScheduleArray.add(scheduleMap);
      }
     }
    }
   }
  }
  return paymentScheduleArray;
 }
}
```

- Per i dettagli sull'API, consulta la sezione GetAgreementTerms AWS SDK for Java 2.xAPI Reference.

Ottieni i prezzi per dimensione in un contratto

Il seguente esempio di codice mostra come ottenere il prezzo per dimensione in un accordo.

SDK per Java 2.x

> **Note**
>
> C'è altro da sapere GitHub. Trova l'esempio completo e scopri come configurarlo ed eseguirlo nel repository Marketplace AWS API Reference Code Library.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package com.example.awsmarketplace.agreementapi;
```

```
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import software.amazon.awssdk.services.marketplaceagreement.model.AcceptedTerm;
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementTermsRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementTermsResponse;

import java.util.ArrayList;
import java.util.List;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.AGREEMENT_ID;
import com.example.awsmarketplace.utils.ReferenceCodesUtils;

public class GetAgreementTermsPricingEachDimension {

 /*
  * Obtain pricing per each dimension in the agreement
  */
 public static void main(String[] args) {

  String agreementId = args.length > 0 ? args[0] : AGREEMENT_ID;

  List<Object> dimensions = getDimensions(agreementId);

  ReferenceCodesUtils.formatOutput(dimensions);
 }

 public static List<Object> getDimensions(String agreementId) {
  MarketplaceAgreementClient marketplaceAgreementClient =
    MarketplaceAgreementClient.builder()
    .httpClient(ApacheHttpClient.builder().build())
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

  GetAgreementTermsRequest getAgreementTermsRequest =
    GetAgreementTermsRequest.builder().agreementId(agreementId)
    .build();

  GetAgreementTermsResponse getAgreementTermsResponse =
 marketplaceAgreementClient.getAgreementTerms(getAgreementTermsRequest);
```

```
  List<Object> dimensions = new ArrayList<Object>();

  for (AcceptedTerm acceptedTerm : getAgreementTermsResponse.acceptedTerms()) {
   List<Object> rateInfo = new ArrayList<Object>();
   if (acceptedTerm.configurableUpfrontPricingTerm() != null) {
    if (acceptedTerm.configurableUpfrontPricingTerm().type() != null) {
     rateInfo.add(acceptedTerm.configurableUpfrontPricingTerm().type());
    }
    if (acceptedTerm.configurableUpfrontPricingTerm().currencyCode() != null) {
     rateInfo.add(acceptedTerm.configurableUpfrontPricingTerm().currencyCode());
    }
    if (acceptedTerm.configurableUpfrontPricingTerm().hasRateCards()) {
     rateInfo.add(acceptedTerm.configurableUpfrontPricingTerm().rateCards());
    }
    dimensions.add(rateInfo);
   }
  }
  return dimensions;
 }
}
```

- Per i dettagli sull'API, consulta la sezione GetAgreementTerms AWS SDK for Java 2.xAPI Reference.

Scopri il tipo di prezzo di un contratto

Il seguente esempio di codice mostra come ottenere il tipo di prezzo di un contratto.

SDK per Java 2.x

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri come configurarlo ed eseguirlo nel repository Marketplace AWS API Reference Code Library.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package com.example.awsmarketplace.agreementapi;
```

```
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import software.amazon.awssdk.services.marketplaceagreement.model.AcceptedTerm;
import
 software.amazon.awssdk.services.marketplaceagreement.model.AgreementViewSummary;
import software.amazon.awssdk.services.marketplaceagreement.model.Filter;
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementTermsRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementTermsResponse;
import
 software.amazon.awssdk.services.marketplaceagreement.model.SearchAgreementsRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.SearchAgreementsResponse;

import com.fasterxml.jackson.annotation.JsonAutoDetect.Visibility;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.HashMap;
import java.util.HashSet;
import java.util.Iterator;
import java.util.List;
import java.util.Map;
import java.util.Objects;
import java.util.Set;

import org.apache.commons.lang3.tuple.Triple;

import software.amazon.awssdk.services.marketplacecatalog.MarketplaceCatalogClient;
import
 software.amazon.awssdk.services.marketplacecatalog.model.DescribeEntityRequest;
import
 software.amazon.awssdk.services.marketplacecatalog.model.DescribeEntityResponse;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.*;
import com.fasterxml.jackson.annotation.PropertyAccessor;
import com.fasterxml.jackson.core.JsonProcessingException;
import com.fasterxml.jackson.databind.JsonNode;
import com.fasterxml.jackson.databind.ObjectMapper;
import com.fasterxml.jackson.databind.ObjectWriter;
import com.fasterxml.jackson.datatype.jsr310.JavaTimeModule;
```

```
/*
 * Obtain the pricing type of the agreement (contract, FPS, metered, free etc.)
 */
public class GetAgreementPricingType {

 private static final String FILTER_NAME = "OfferId";

 private static final String FILTER_VALUE = OFFER_ID;

 // Product types
 private static final String SAAS_PRODUCT = "SaaSProduct";
 private static final String AMI_PRODUCT = "AmiProduct";
 private static final String ML_PRODUCT = "MachineLearningProduct";
 private static final String CONTAINER_PRODUCT = "ContainerProduct";
 private static final String DATA_PRODUCT = "DataProduct";
 private static final String PROSERVICE_PRODUCT = "ProfessionalServicesProduct";
 private static final String AIQ_PRODUCT = "AiqProduct";

 // Pricing types
 private static final String CCP = "CCP";
 private static final String ANNUAL = "Annual";
 private static final String CONTRACT = "Contract";
 private static final String SFT = "SaaS Free Trial";
 private static final String HMA = "Hourly and Monthly Agreements";
 private static final String HOURLY = "Hourly";
 private static final String MONTHLY = "Monthly";
 private static final String AFPS = "Annual FPS";
 private static final String CFPS = "Contract FPS";
 private static final String CCPFPS = "CCP with FPS";
 private static final String BYOL = "BYOL";
 private static final String FREE = "Free";
 private static final String FTH = "Free Trials and Hourly";

 // Agreement term pricing types
 private static final Set<String> LEGAL = Set.of("LegalTerm");
 private static final Set<String> CONFIGURABLE_UPFRONT =
 Set.of("ConfigurableUpfrontPricingTerm");
 private static final Set<String> USAGE_BASED = Set.of("UsageBasedPricingTerm");
 private static final Set<String> CONFIGURABLE_UPFRONT_AND_USAGE_BASED =
 Set.of("ConfigurableUpfrontPricingTerm", "UsageBasedPricingTerm");
 private static final Set<String> FREE_TRIAL = Set.of("FreeTrialPricingTerm");
 private static final Set<String> RECURRING_PAYMENT =
 Set.of("RecurringPaymentTerm");
```

```
private static final Set<String> USAGE_BASED_AND_RECURRING_PAYMENT =
Set.of("UsageBasedPricingTerm", "RecurringPaymentTerm");
private static final Set<String> FIXED_UPFRONT_AND_PAYMENT_SCHEDULE =
Set.of("FixedUpfrontPricingTerm", "PaymentScheduleTerm");
private static final Set<String> FIXED_UPFRONT_AND_PAYMENT_SCHEDULE_AND_USAGE_BASED
= Set.of("FixedUpfrontPricingTerm", "PaymentScheduleTerm",
"UsageBasedPricingTerm");
private static final Set<String> BYOL_PRICING = Set.of("ByolPricingTerm");
private static final Set<String> FREE_TRIAL_AND_USAGE_BASED =
Set.of("FreeTrialPricingTerm", "UsageBasedPricingTerm");

private static final List<Set<String>> ALL_AGREEMENT_TERM_TYPES_COMBINATION
= Arrays.asList(LEGAL, CONFIGURABLE_UPFRONT, USAGE_BASED,
CONFIGURABLE_UPFRONT_AND_USAGE_BASED,
  FREE_TRIAL, RECURRING_PAYMENT, USAGE_BASED_AND_RECURRING_PAYMENT,
FIXED_UPFRONT_AND_PAYMENT_SCHEDULE,
FIXED_UPFRONT_AND_PAYMENT_SCHEDULE_AND_USAGE_BASED, BYOL_PRICING,
FREE_TRIAL_AND_USAGE_BASED);

private static  MarketplaceAgreementClient marketplaceAgreementClient =
  MarketplaceAgreementClient.builder()
  .httpClient(ApacheHttpClient.builder().build())
  .credentialsProvider(ProfileCredentialsProvider.create())
  .build();

private static MarketplaceCatalogClient marketplaceCatalogClient =
  MarketplaceCatalogClient.builder()
  .httpClient(ApacheHttpClient.builder().build())
  .credentialsProvider(ProfileCredentialsProvider.create())
  .build();

   /*
    * Get agreement Pricing Type given product type, agreement term types and offer
types if needed
    */
public static String getPricingType(String productType, Set<String>
agreementTermType, Set<String> offerType) {
 Map<Triple<String, Set<String>, Set<String>>, String> pricingTypes = new
HashMap<>();

 pricingTypes.put(Triple.of(SAAS_PRODUCT, CONFIGURABLE_UPFRONT_AND_USAGE_BASED, new
HashSet<>()), CCP);
 pricingTypes.put(Triple.of(DATA_PRODUCT, CONFIGURABLE_UPFRONT_AND_USAGE_BASED, new
HashSet<>()), CCP);
```

```
 pricingTypes.put(Triple.of(CONTAINER_PRODUCT, CONFIGURABLE_UPFRONT,
CONFIGURABLE_UPFRONT_AND_USAGE_BASED), ANNUAL);
 pricingTypes.put(Triple.of(AMI_PRODUCT, CONFIGURABLE_UPFRONT,
CONFIGURABLE_UPFRONT_AND_USAGE_BASED), ANNUAL);
 pricingTypes.put(Triple.of(ML_PRODUCT, CONFIGURABLE_UPFRONT,
CONFIGURABLE_UPFRONT_AND_USAGE_BASED), ANNUAL);
 pricingTypes.put(Triple.of(CONTAINER_PRODUCT, CONFIGURABLE_UPFRONT,
CONFIGURABLE_UPFRONT), CONTRACT);
 pricingTypes.put(Triple.of(AMI_PRODUCT, CONFIGURABLE_UPFRONT,
CONFIGURABLE_UPFRONT), CONTRACT);
 pricingTypes.put(Triple.of(SAAS_PRODUCT, CONFIGURABLE_UPFRONT, new HashSet<>()),
CONTRACT);
 pricingTypes.put(Triple.of(DATA_PRODUCT, CONFIGURABLE_UPFRONT, new HashSet<>()),
CONTRACT);
 pricingTypes.put(Triple.of(AIQ_PRODUCT, CONFIGURABLE_UPFRONT, new HashSet<>()),
CONTRACT);
 pricingTypes.put(Triple.of(PROSERVICE_PRODUCT, CONFIGURABLE_UPFRONT, new
HashSet<>()), CONTRACT);
 pricingTypes.put(Triple.of(SAAS_PRODUCT, FREE_TRIAL, new HashSet<>()), SFT);
 pricingTypes.put(Triple.of(AMI_PRODUCT, USAGE_BASED_AND_RECURRING_PAYMENT, new
HashSet<>()), HMA);
 pricingTypes.put(Triple.of(SAAS_PRODUCT, USAGE_BASED, new HashSet<>()), HOURLY);
 pricingTypes.put(Triple.of(AMI_PRODUCT, USAGE_BASED, new HashSet<>()), HOURLY);
 pricingTypes.put(Triple.of(ML_PRODUCT, USAGE_BASED, new HashSet<>()), HOURLY);
 pricingTypes.put(Triple.of(CONTAINER_PRODUCT, RECURRING_PAYMENT, new HashSet<>()),
MONTHLY);
 pricingTypes.put(Triple.of(AMI_PRODUCT, RECURRING_PAYMENT, new HashSet<>()),
MONTHLY);
 pricingTypes.put(Triple.of(CONTAINER_PRODUCT, FIXED_UPFRONT_AND_PAYMENT_SCHEDULE,
FIXED_UPFRONT_AND_PAYMENT_SCHEDULE_AND_USAGE_BASED), AFPS);
 pricingTypes.put(Triple.of(AMI_PRODUCT, FIXED_UPFRONT_AND_PAYMENT_SCHEDULE,
FIXED_UPFRONT_AND_PAYMENT_SCHEDULE_AND_USAGE_BASED), AFPS);
 pricingTypes.put(Triple.of(ML_PRODUCT, FIXED_UPFRONT_AND_PAYMENT_SCHEDULE, new
HashSet<>()), AFPS);
 pricingTypes.put(Triple.of(CONTAINER_PRODUCT, FIXED_UPFRONT_AND_PAYMENT_SCHEDULE,
new HashSet<>()), CFPS);
 pricingTypes.put(Triple.of(AMI_PRODUCT, FIXED_UPFRONT_AND_PAYMENT_SCHEDULE,
FIXED_UPFRONT_AND_PAYMENT_SCHEDULE), CFPS);
 pricingTypes.put(Triple.of(SAAS_PRODUCT, FIXED_UPFRONT_AND_PAYMENT_SCHEDULE, new
HashSet<>()), CFPS);
 pricingTypes.put(Triple.of(DATA_PRODUCT, FIXED_UPFRONT_AND_PAYMENT_SCHEDULE, new
HashSet<>()), CFPS);
 pricingTypes.put(Triple.of(AIQ_PRODUCT, FIXED_UPFRONT_AND_PAYMENT_SCHEDULE, new
HashSet<>()), CFPS);
```

```
 pricingTypes.put(Triple.of(PROSERVICE_PRODUCT, FIXED_UPFRONT_AND_PAYMENT_SCHEDULE,
new HashSet<>()), CFPS);
 pricingTypes.put(Triple.of(SAAS_PRODUCT,
FIXED_UPFRONT_AND_PAYMENT_SCHEDULE_AND_USAGE_BASED, new HashSet<>()), CCPFPS);
 pricingTypes.put(Triple.of(DATA_PRODUCT,
FIXED_UPFRONT_AND_PAYMENT_SCHEDULE_AND_USAGE_BASED, new HashSet<>()), CCPFPS);
 pricingTypes.put(Triple.of(AIQ_PRODUCT,
FIXED_UPFRONT_AND_PAYMENT_SCHEDULE_AND_USAGE_BASED, new HashSet<>()), CCPFPS);
 pricingTypes.put(Triple.of(PROSERVICE_PRODUCT,
FIXED_UPFRONT_AND_PAYMENT_SCHEDULE_AND_USAGE_BASED, new HashSet<>()), CCPFPS);
 pricingTypes.put(Triple.of(AMI_PRODUCT, BYOL_PRICING, new HashSet<>()), BYOL);
 pricingTypes.put(Triple.of(SAAS_PRODUCT, BYOL_PRICING, new HashSet<>()), BYOL);
 pricingTypes.put(Triple.of(PROSERVICE_PRODUCT, BYOL_PRICING, new HashSet<>()),
BYOL);
 pricingTypes.put(Triple.of(AIQ_PRODUCT, BYOL_PRICING, new HashSet<>()), BYOL);
 pricingTypes.put(Triple.of(ML_PRODUCT, BYOL_PRICING, new HashSet<>()), BYOL);
 pricingTypes.put(Triple.of(CONTAINER_PRODUCT, BYOL_PRICING, new HashSet<>()),
BYOL);
 pricingTypes.put(Triple.of(DATA_PRODUCT, BYOL_PRICING, new HashSet<>()), BYOL);
 pricingTypes.put(Triple.of(CONTAINER_PRODUCT, LEGAL, new HashSet<>()), FREE);
 pricingTypes.put(Triple.of(AMI_PRODUCT, FREE_TRIAL_AND_USAGE_BASED, new
HashSet<>()), FTH);
 pricingTypes.put(Triple.of(CONTAINER_PRODUCT, FREE_TRIAL_AND_USAGE_BASED, new
HashSet<>()), FTH);
 pricingTypes.put(Triple.of(ML_PRODUCT, FREE_TRIAL_AND_USAGE_BASED, new
HashSet<>()), FTH);

 Triple<String, Set<String>, Set<String>> key = Triple.of(productType,
agreementTermType, offerType);

 if (pricingTypes.containsKey(key)) {
  return pricingTypes.get(key);
 } else {
  return "Unknown";
 }
}

/*
 * Given product type and agreement term types, some combinations need to check
offer term types as well.
 */
public static String needToCheckOfferTermsType(String productType, Set<String>
agreementTermTypes) {
 Map<KeyPair, String> offerTermTypes = new HashMap<>();
```

```
 offerTermTypes.put(new KeyPair(CONTAINER_PRODUCT, CONFIGURABLE_UPFRONT), "Y");
 offerTermTypes.put(new KeyPair(AMI_PRODUCT, CONFIGURABLE_UPFRONT), "Y");
 offerTermTypes.put(new KeyPair(CONTAINER_PRODUCT,
FIXED_UPFRONT_AND_PAYMENT_SCHEDULE), "Y");
 offerTermTypes.put(new KeyPair(AMI_PRODUCT, FIXED_UPFRONT_AND_PAYMENT_SCHEDULE),
"Y");

 KeyPair key = new KeyPair(productType, agreementTermTypes);
 if (offerTermTypes.containsKey(key)) {
  return offerTermTypes.get(key);
 } else {
  return null;
 }
}

public static List<AgreementViewSummary> getAgreementsById() {

 List<AgreementViewSummary> agreementSummaryList = new
ArrayList<AgreementViewSummary>();

 Filter partyType =
Filter.builder().name(PARTY_TYPE_FILTER_NAME).values(PARTY_TYPE_FILTER_VALUE_PROPOSER).buil

 Filter agreementType =
Filter.builder().name(AGREEMENT_TYPE_FILTER_NAME).values(AGREEMENT_TYPE_FILTER_VALUE_PURCHA

 Filter customizeFilter =
Filter.builder().name(FILTER_NAME).values(FILTER_VALUE).build();

 SearchAgreementsRequest searchAgreementsRequest =
   SearchAgreementsRequest.builder()
   .catalog(AWS_MP_CATALOG)
   .filters(partyType, agreementType, customizeFilter).build();

 SearchAgreementsResponse searchResultResponse =
marketplaceAgreementClient.searchAgreements(searchAgreementsRequest);

 agreementSummaryList.addAll(searchResultResponse.agreementViewSummaries());

 while (searchResultResponse.nextToken() != null &&
searchResultResponse.nextToken().length() > 0) {
  searchAgreementsRequest =
SearchAgreementsRequest.builder().catalog(AWS_MP_CATALOG)
```

```
    .filters(partyType,
agreementType).nextToken(searchResultResponse.nextToken()).build();
  searchResultResponse =
marketplaceAgreementClient.searchAgreements(searchAgreementsRequest);
  agreementSummaryList.addAll(searchResultResponse.agreementViewSummaries());
 }
 return agreementSummaryList;

}

static class KeyPair {
 private final String first;
 private final Set<String> second;

 public KeyPair(String productType, Set<String> second) {
  this.first = productType;
  this.second = second;
 }

 @Override
 public int hashCode() {
  return Objects.hash(first, second);
 }

 @Override
 public boolean equals(Object obj) {
  if (this == obj)
   return true;
  if (obj == null || getClass() != obj.getClass())
   return false;
  KeyPair other = (KeyPair) obj;
  return Objects.equals(first, other.first) && Objects.equals(second,
other.second);
 }
}

/*
 * Get all the term types for the offer
 */
public static Set<String> getOfferTermTypes(String offerId) {

 Set<String> offerTermTypes = new HashSet<String>();

 DescribeEntityRequest request =
```

```
    DescribeEntityRequest.builder()
    .catalog(AWS_MP_CATALOG)
    .entityId(offerId)
    .build();

 DescribeEntityResponse result = marketplaceCatalogClient.describeEntity(request);

 String details = result.details();

 try {
  ObjectMapper objectMapper = new ObjectMapper();
  JsonNode rootNode = objectMapper.readTree(details);
  JsonNode termsNode = rootNode.get(ATTRIBUTE_TERMS);

  for (JsonNode termNode : termsNode) {
   if (termNode.get(ATTRIBUTE_TYPE_ENTITY) != null ) {
    offerTermTypes.add(termNode.get(ATTRIBUTE_TYPE_ENTITY).asText());
   }
  }
 } catch (Exception e) {
  e.printStackTrace();
 }

 return offerTermTypes;

}

/*
 * Get all the agreement term types
 */
public static Set<String> getAgreementTermTypes(GetAgreementTermsResponse
agreementTerm) {
 Set<String> agreementTermTypes = new HashSet<String>();
 try {
  for (AcceptedTerm term : agreementTerm.acceptedTerms()) {
   ObjectMapper objectMapper = new ObjectMapper();
   JsonNode termNode = objectMapper.readTree(getJson(term));
   Iterator<Map.Entry<String, JsonNode>> fieldsIterator = termNode.fields();
   while (fieldsIterator.hasNext()) {
    Map.Entry<String, JsonNode> entry = fieldsIterator.next();
    JsonNode value = entry.getValue();
    if (value.isObject() && value.has(ATTRIBUTE_TYPE_AGREEMENT)) {
     agreementTermTypes.add(value.get(ATTRIBUTE_TYPE_AGREEMENT).asText());
    }
```

```
    }
   }
  } catch (Exception e) {
   e.printStackTrace();
  }
  return agreementTermTypes;

}

/*
 * make sure all elements in array2 exist in array1
 */
public static boolean allElementsExist(Set<String> array1, Set<String> array2) {
 for (String element : array2) {
  boolean found = false;
  for (String str : array1) {
   if (element.equals(str)) {
    found = true;
    break;
   }
  }
  if (!found) {
   return false;
  }
 }
 return true;
}

/*
 * Find the combinations of the agreement term types for the agreement
 */
public static Set<String> getMatchedTermTypesCombination(Set<String>
agreementTermTypes) {
 Set<String> matchedCombination = new HashSet<String>();
 for (Set<String> element : ALL_AGREEMENT_TERM_TYPES_COMBINATION) {
  if (allElementsExist(agreementTermTypes, element)) {
   matchedCombination = element;
  }
 }
 return matchedCombination;
}

public static void main(String[] args) {
```

```
 List<AgreementViewSummary> agreements = getAgreementsById();

 for (AgreementViewSummary summary : agreements) {
  String pricingType = "";
  String agreementId = summary.agreementId();
  System.out.println(agreementId);
  String offerId = summary.proposalSummary().offerId();

  //get all pricing term types for the offer in the agreement
  Set<String> offerTermTypes = getOfferTermTypes(offerId);
  String productType = summary.proposalSummary().resources().get(0).type();

  //get all pricing term types for the agreement
  GetAgreementTermsRequest getAgreementTermsRequest =
    GetAgreementTermsRequest.builder().agreementId(agreementId)
    .build();
  GetAgreementTermsResponse getAgreementTermsResponse =
marketplaceAgreementClient.getAgreementTerms(getAgreementTermsRequest);
  Set<String> agreementTermTypes =
getAgreementTermTypes(getAgreementTermsResponse);

  //get matched pricing term type combination set
  Set<String> agreementMatchedTermType =
getMatchedTermTypesCombination(agreementTermTypes);

  //check to see if this agreement pricing term combination needs additional check
on offer pricing terms
  String needToCheckOfferType = needToCheckOfferTermsType(productType,
agreementMatchedTermType);

  // get the pricing type for the agreement based on the product type, agreement
term types and offer term types if needed
  if (needToCheckOfferType != null) {
   Set<String> offerMatchedTermType =
getMatchedTermTypesCombination(offerTermTypes);
   pricingType = getPricingType(productType, agreementMatchedTermType,
offerMatchedTermType);
  } else if (agreementMatchedTermType == LEGAL) {
   pricingType = FREE;
  } else {
   pricingType = getPricingType(productType, agreementMatchedTermType, new
HashSet());
  }
  System.out.println("Pricing type is " + pricingType);
```

```
  }
 }

 private static String getJson(Object result) {
  String json = "";

  try {
   ObjectMapper om = new ObjectMapper();
   om.setVisibility(PropertyAccessor.FIELD, Visibility.ANY);
   om.registerModule(new JavaTimeModule());
   ObjectWriter ow = om.writer().withDefaultPrettyPrinter();

   json = ow.writeValueAsString(result);
  } catch (JsonProcessingException e) {
   e.printStackTrace();
  }
  return json;
 }

}
```

- Per i dettagli sull'API, consulta la sezione DescribeAgreement AWS SDK for Java 2.xAPI Reference.

Ottieni il tipo di prodotto di un contratto

Il seguente esempio di codice mostra come ottenere il tipo di prodotto di un contratto.

SDK per Java 2.x

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri come configurarlo ed eseguirlo nel repository Marketplace AWS API Reference Code Library.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package com.example.awsmarketplace.agreementapi;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
```

```
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import
 software.amazon.awssdk.services.marketplaceagreement.model.DescribeAgreementRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.DescribeAgreementResponse;
import software.amazon.awssdk.services.marketplaceagreement.model.Resource;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.*;

import java.util.ArrayList;
import java.util.List;

import com.example.awsmarketplace.utils.ReferenceCodesUtils;

public class GetAgreementProductType {

 /*
  * Obtain the Product Type of the product the agreement was created on
  */
 public static void main(String[] args) {

  String agreementId = args.length > 0 ? args[0] : AGREEMENT_ID;

  List<String> productIds = getProducts(agreementId);

  ReferenceCodesUtils.formatOutput(productIds);
 }

 public static List<String> getProducts(String agreementId) {
  MarketplaceAgreementClient marketplaceAgreementClient =
    MarketplaceAgreementClient.builder()
    .httpClient(ApacheHttpClient.builder().build())
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

  DescribeAgreementRequest describeAgreementRequest =
    DescribeAgreementRequest.builder()
    .agreementId(agreementId)
    .build();

  DescribeAgreementResponse describeAgreementResponse =
 marketplaceAgreementClient.describeAgreement(describeAgreementRequest);
```

```
  List<String> productIds = new ArrayList<String>();
  for (Resource resource : describeAgreementResponse.proposalSummary().resources())
 {
   productIds.add(resource.id() + ":" + resource.type());
  }
  return productIds;
 }
}
```

- Per i dettagli sull'API, consulta la sezione DescribeAgreement AWS SDK for Java 2.xAPI Reference.

Ottieni lo stato di un accordo

Il seguente esempio di codice mostra come ottenere lo stato di un accordo.

SDK per Java 2.x

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri come configurarlo ed eseguirlo nel repository Marketplace AWS API Reference Code Library.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package com.example.awsmarketplace.agreementapi;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.AGREEMENT_ID;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import
 software.amazon.awssdk.services.marketplaceagreement.model.DescribeAgreementRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.DescribeAgreementResponse;
```

```
public class GetAgreementStatus {

 public static void main(String[] args) {

  String agreementId = args.length > 0 ? args[0] : AGREEMENT_ID;

  DescribeAgreementResponse describeAgreementResponse =
 getDescribeAgreementResponse(agreementId);

  System.out.println("Agreement status is " + describeAgreementResponse.status());

 }

 public static DescribeAgreementResponse getDescribeAgreementResponse(String
 agreementId) {
  MarketplaceAgreementClient marketplaceAgreementClient =
    MarketplaceAgreementClient.builder()
    .httpClient(ApacheHttpClient.builder().build())
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

  DescribeAgreementRequest describeAgreementRequest =
    DescribeAgreementRequest.builder()
    .agreementId(agreementId)
    .build();

  DescribeAgreementResponse describeAgreementResponse =
 marketplaceAgreementClient.describeAgreement(describeAgreementRequest);
  return describeAgreementResponse;
 }

}
```

- Per i dettagli sull'API, consulta la sezione DescribeAgreement AWS SDK for Java 2.xAPI Reference.

Consulta i termini di supporto di un contratto

Il seguente esempio di codice mostra come ottenere i termini di assistenza di un contratto.

SDK per Java 2.x

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri come configurarlo ed eseguirlo nel repository Marketplace AWS API Reference Code Library.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package com.example.awsmarketplace.agreementapi;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import software.amazon.awssdk.services.marketplaceagreement.model.AcceptedTerm;
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementTermsRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementTermsResponse;
import software.amazon.awssdk.services.marketplaceagreement.model.SupportTerm;

import java.util.ArrayList;
import java.util.List;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.AGREEMENT_ID;
import com.example.awsmarketplace.utils.ReferenceCodesUtils;

public class GetAgreementTermsSupportTerm {

 /*
  * Obtain the support and refund policy I have provided to the customer
  */
 public static void main(String[] args) {

  String agreementId = args.length > 0 ? args[0] : AGREEMENT_ID;

  List<SupportTerm> supportTerms = getSupportTerms(agreementId);

  ReferenceCodesUtils.formatOutput(supportTerms);
 }
```

```
 public static List<SupportTerm> getSupportTerms(String agreementId) {
  MarketplaceAgreementClient marketplaceAgreementClient =
    MarketplaceAgreementClient.builder()
    .httpClient(ApacheHttpClient.builder().build())
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

  GetAgreementTermsRequest getAgreementTermsRequest =
    GetAgreementTermsRequest.builder().agreementId(agreementId)
    .build();

  GetAgreementTermsResponse getAgreementTermsResponse =
 marketplaceAgreementClient.getAgreementTerms(getAgreementTermsRequest);

  List<SupportTerm> supportTerms = new ArrayList<>();

  for (AcceptedTerm acceptedTerm : getAgreementTermsResponse.acceptedTerms()) {
   if (acceptedTerm.supportTerm() != null) {
    supportTerms.add(acceptedTerm.supportTerm());
   }
  }
  return supportTerms;
 }

}
```

- Per i dettagli sull'API, consulta la sezione GetAgreementTerms AWS SDK for Java 2.xAPI Reference.

Consulta i termini di un contratto

Il seguente esempio di codice mostra come ottenere i termini di un accordo.

SDK per Java 2.x

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri come configurarlo ed eseguirlo nel repository Marketplace AWS API Reference Code Library.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package com.example.awsmarketplace.agreementapi;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.*;
import com.example.awsmarketplace.utils.ReferenceCodesUtils;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementTermsRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementTermsResponse;

public class GetAgreementTerms {

 public static void main(String[] args) {

  String agreementId = args.length > 0 ? args[0] : AGREEMENT_ID;

  GetAgreementTermsResponse getAgreementTermsResponse =
 getAgreementTermsResponse(agreementId);

  ReferenceCodesUtils.formatOutput(getAgreementTermsResponse);

 }

 public static GetAgreementTermsResponse getAgreementTermsResponse(String
 agreementId) {
  MarketplaceAgreementClient marketplaceAgreementClient =
    MarketplaceAgreementClient.builder()
    .httpClient(ApacheHttpClient.builder().build())
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

  GetAgreementTermsRequest getAgreementTermsRequest =
    GetAgreementTermsRequest.builder()
    .agreementId(agreementId)
    .build();
```

```
  GetAgreementTermsResponse getAgreementTermsResponse =
 marketplaceAgreementClient.getAgreementTerms(getAgreementTermsRequest);
  return getAgreementTermsResponse;
 }

}
```

- Per i dettagli sull'API, consulta la sezione GetAgreementTerms AWS SDK for Java 2.xAPI Reference.

Cerca gli accordi per data di fine

Il seguente esempio di codice mostra come cercare gli accordi per data di fine.

SDK per Java 2.x

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri come configurarlo ed eseguirlo nel repository Marketplace AWS API Reference Code Library.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package com.example.awsmarketplace.agreementapi;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import
 software.amazon.awssdk.services.marketplaceagreement.model.AgreementViewSummary;
import software.amazon.awssdk.services.marketplaceagreement.model.Filter;
import
 software.amazon.awssdk.services.marketplaceagreement.model.SearchAgreementsRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.SearchAgreementsResponse;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;
```

```
import static com.example.awsmarketplace.utils.ReferenceCodesConstants.*;
import com.example.awsmarketplace.utils.ReferenceCodesUtils;

public class SearchAgreementsByEndDate {

 static String beforeOrAfterEndtimeFilterName =
 BeforeOrAfterEndTimeFilterName.BeforeEndTime.name();

 static String cutoffDate = "2050-11-18T00:00:00Z";

 static String partyTypeFilterValue = PARTY_TYPE_FILTER_VALUE_PROPOSER;

 public static void main(String[] args) {

  List<AgreementViewSummary> agreementSummaryList = getAgreements();

  ReferenceCodesUtils.formatOutput(agreementSummaryList);
 }

 public static List<AgreementViewSummary> getAgreements() {
  MarketplaceAgreementClient marketplaceAgreementClient =
    MarketplaceAgreementClient.builder()
    .httpClient(ApacheHttpClient.builder().build())
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

  // set up filters

  Filter partyTypeFilter = Filter.builder().name(PARTY_TYPE_FILTER_NAME)
    .values(PARTY_TYPE_FILTER_VALUE_PROPOSER).build();

  Filter agreementTypeFilter = Filter.builder().name(AGREEMENT_TYPE_FILTER_NAME)
    .values(AGREEMENT_TYPE_FILTER_VALUE_PURCHASEAGREEMENT).build();

  Filter customizeFilter =
 Filter.builder().name(beforeOrAfterEndtimeFilterName).values(cutoffDate).build();

  List<Filter> filters = new ArrayList<Filter>();

  filters.addAll(Arrays.asList(partyTypeFilter, agreementTypeFilter,
 customizeFilter));

  // search agreement with filters
```

```
 SearchAgreementsRequest searchAgreementsRequest =
   SearchAgreementsRequest.builder()
   .catalog(AWS_MP_CATALOG)
   .filters(filters)
   .build();

 SearchAgreementsResponse searchAgreementResponse=
marketplaceAgreementClient.searchAgreements(searchAgreementsRequest);

 List<AgreementViewSummary> agreementSummaryList = new
ArrayList<AgreementViewSummary>();

 agreementSummaryList.addAll(searchAgreementResponse.agreementViewSummaries());

 while (searchAgreementResponse.nextToken() != null &&
searchAgreementResponse.nextToken().length() > 0) {
  searchAgreementsRequest =
    SearchAgreementsRequest.builder()
    .catalog(AWS_MP_CATALOG)
    .filters(filters)
    .nextToken(searchAgreementResponse.nextToken())
    .build();
  searchAgreementResponse =
marketplaceAgreementClient.searchAgreements(searchAgreementsRequest);
  agreementSummaryList.addAll(searchAgreementResponse.agreementViewSummaries());
 }
 return agreementSummaryList;
}

}
```

- Per i dettagli sull'API, consulta la sezione SearchAgreements AWS SDK for Java 2.xAPI Reference.

Cerca gli accordi con un filtro personalizzato

Il seguente esempio di codice mostra come cercare accordi con un filtro personalizzato.

SDK per Java 2.x

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri come configurarlo ed eseguirlo nel repository Marketplace AWS API Reference Code Library.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package com.example.awsmarketplace.agreementapi;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import
 software.amazon.awssdk.services.marketplaceagreement.model.AgreementViewSummary;
import software.amazon.awssdk.services.marketplaceagreement.model.Filter;
import
 software.amazon.awssdk.services.marketplaceagreement.model.SearchAgreementsRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.SearchAgreementsResponse;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.*;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;

import com.example.awsmarketplace.utils.ReferenceCodesUtils;

/**
 * To search by
 * offer id: OfferId;
 * product id: ResourceIdentifier;
 * customer AWS account id: AcceptorAccountId
 * product type: ResourceType (i.e. SaasProduct)
 * status: Status. status values can be: ACTIVE, CANCELED,
 *   EXPIRED, RENEWED, REPLACED, ROLLED_BACK, SUPERSEDED, TERMINATED
 */
```

```
public class SearchAgreementsByOneFilter {

 private static final String FILTER_NAME = "ResourceType";

 private static final String FILTER_VALUE = "SaaSProduct";

 /*
  * search agreements by one customize filter
  */
 public static void main(String[] args) {

  List<AgreementViewSummary> agreementSummaryList = getAgreements();

  ReferenceCodesUtils.formatOutput(agreementSummaryList);
 }

 public static List<AgreementViewSummary> getAgreements() {
  MarketplaceAgreementClient marketplaceAgreementClient =
    MarketplaceAgreementClient.builder()
    .httpClient(ApacheHttpClient.builder().build())
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

  Filter partyTypeFilter = Filter.builder().name(PARTY_TYPE_FILTER_NAME)
    .values(PARTY_TYPE_FILTER_VALUE_PROPOSER).build();

  Filter agreementTypeFilter = Filter.builder().name(AGREEMENT_TYPE_FILTER_NAME)
    .values(AGREEMENT_TYPE_FILTER_VALUE_PURCHASEAGREEMENT).build();

  Filter customizeFilter =
Filter.builder().name(FILTER_NAME).values(FILTER_VALUE).build();

  List<Filter> filters = new ArrayList<Filter>();

  filters.addAll(Arrays.asList(partyTypeFilter, agreementTypeFilter,
customizeFilter));

  SearchAgreementsRequest searchAgreementsRequest =
    SearchAgreementsRequest.builder()
    .catalog(AWS_MP_CATALOG)
    .filters(filters)
    .build();
  SearchAgreementsResponse searchAgreementsResponse =
marketplaceAgreementClient.searchAgreements(searchAgreementsRequest);
```

```
 List<AgreementViewSummary> agreementSummaryList = new
ArrayList<AgreementViewSummary>();

 agreementSummaryList.addAll(searchAgreementsResponse.agreementViewSummaries());

 while (searchAgreementsResponse.nextToken() != null &&
searchAgreementsResponse.nextToken().length() > 0) {
  searchAgreementsRequest =
    SearchAgreementsRequest.builder()
    .catalog(AWS_MP_CATALOG)
    .filters(filters)
    .nextToken(searchAgreementsResponse.nextToken())
    .build();
  searchAgreementsResponse =
marketplaceAgreementClient.searchAgreements(searchAgreementsRequest);
  agreementSummaryList.addAll(searchAgreementsResponse.agreementViewSummaries());
 }
 return agreementSummaryList;
}

}
```

- Per i dettagli sull'API, consulta la sezione SearchAgreements AWS SDK for Java 2.xAPI Reference.

Cerca gli accordi con due filtri personalizzati

Il seguente esempio di codice mostra come cercare accordi con due filtri personalizzati.

SDK per Java 2.x

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri come configurarlo ed eseguirlo nel repository Marketplace AWS API Reference Code Library.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package com.example.awsmarketplace.agreementapi;
```

```
import software.amazon.awssdk.auth.credentials.AwsCredentialsProvider;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.SdkHttpClient;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import
 software.amazon.awssdk.services.marketplaceagreement.model.AgreementViewSummary;
import software.amazon.awssdk.services.marketplaceagreement.model.Filter;
import
 software.amazon.awssdk.services.marketplaceagreement.model.SearchAgreementsRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.SearchAgreementsResponse;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.*;
import com.example.awsmarketplace.utils.ReferenceCodesUtils;

/**
 * Party Type = Proposer AND Acceptor:
 *  AfterEndTime
 *  BeforeEndTime
 *  ResourceIdentifier + BeforeEndTime
 *  ResourceIdentifier + AfterEndTime
 *  ResourceType + BeforeEndTime
 *  ResourceType + AfterEndTime
 *
 * Party Type = Proposer
 *  ResourceIdentifier
 *  OfferId
 *  AcceptorAccountId
 *  Status (ACTIVE)
 *  Status (ACTIVE) + ResourceIdentifier
 *  Status (ACTIVE) + AcceptorAccountId
 *  Status (ACTIVE) + OfferId
 *  Status (ACTIVE) + ResourceType
 *  AcceptorAccountId + BeforeEndTime
 *  AcceptorAccountId + AfterEndTime
 *  AcceptorAccountId + AfterEndTime
 *  OfferId + BeforeEndTime
```

```
 *
 * Status values can be: ACTIVE, CANCELLED, EXPIRED, RENEWED, REPLACED, ROLLED_BACK,
 SUPERSEDED, TERMINATED
 */

public class SearchAgreementsByTwoFilters {

 public static final String FILTER_1_NAME = "ResourceType";

 public static final String FILTER_1_VALUE = "SaaSProduct";

 public static final String FILTER_2_NAME = "Status";

 public static final String FILTER_2_VALUE = "ACTIVE";

 /*
  * search agreements by two customize filter
  */
 public static void main(String[] args) {

  List<AgreementViewSummary> agreementSummaryList = getAgreements();

  ReferenceCodesUtils.formatOutput(agreementSummaryList);

 }

 public static List<AgreementViewSummary> getAgreements() {
  MarketplaceAgreementClient marketplaceAgreementClient =
    MarketplaceAgreementClient.builder()
    .httpClient(ApacheHttpClient.builder().build())
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

  Filter partyTypeFilter = Filter.builder().name(PARTY_TYPE_FILTER_NAME)
    .values(PARTY_TYPE_FILTER_VALUE_PROPOSER).build();

  Filter agreementTypeFilter = Filter.builder().name(AGREEMENT_TYPE_FILTER_NAME)
    .values(AGREEMENT_TYPE_FILTER_VALUE_PURCHASEAGREEMENT).build();

  Filter customizeFilter1 =
 Filter.builder().name(FILTER_1_NAME).values(FILTER_1_VALUE).build();

  Filter customizeFilter2 =
 Filter.builder().name(FILTER_2_NAME).values(FILTER_2_VALUE).build();
```

```
 List<Filter> filters = new ArrayList<Filter>();

 filters.addAll(Arrays.asList(partyTypeFilter, agreementTypeFilter,
customizeFilter1, customizeFilter2));

 SearchAgreementsRequest searchAgreementsRequest =
   SearchAgreementsRequest.builder()
   .catalog(AWS_MP_CATALOG)
   .filters(filters)
   .build();

 SearchAgreementsResponse searchAgreementsResponse =
marketplaceAgreementClient.searchAgreements(searchAgreementsRequest);

 List<AgreementViewSummary> agreementSummaryList = new
ArrayList<AgreementViewSummary>();

 agreementSummaryList.addAll(searchAgreementsResponse.agreementViewSummaries());

 while (searchAgreementsResponse.nextToken() != null &&
searchAgreementsResponse.nextToken().length() > 0) {
  searchAgreementsRequest =
    SearchAgreementsRequest.builder()
    .catalog(AWS_MP_CATALOG)
    .filters(filters)
    .nextToken(searchAgreementsResponse.nextToken())
    .build();
  searchAgreementsResponse =
marketplaceAgreementClient.searchAgreements(searchAgreementsRequest);
  agreementSummaryList.addAll(searchAgreementsResponse.agreementViewSummaries());
 }
 return agreementSummaryList;
}

}
```

- Per i dettagli sull'API, consulta la sezione SearchAgreements AWS SDK for Java 2.xAPI Reference.

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.

Ogni esempio include un collegamento al codice sorgente completo, in cui 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())
                                       .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, consulta la CreateJobsezione 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, consulta la GetJobsezione 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, consulta la ListJobssezione 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.

Ogni esempio include un collegamento al codice sorgente completo, in cui 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, consulta la DeleteProgressUpdateStreamsezione 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, consulta la DescribeApplicationStatesezione 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, consulta la DescribeMigrationTasksezione 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, consulta la ImportMigrationTasksezione 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, consulta la ListApplicationssezione 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("ProgressSteamB")
                    .build();

            ListCreatedArtifactsResponse response =
 migrationClient.listCreatedArtifacts(listCreatedArtifactsRequest);
            List<CreatedArtifact> apps = response.createdArtifactList();
            for (CreatedArtifact artifact : apps) {
                System.out.println("APp Id is " + artifact.description());
                System.out.println("The name is " + artifact.name());
            }

        } catch (MigrationHubException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
```

```
}
```

- Per i dettagli sull'API, consulta la ListCreatedArtifactssezione 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, consulta la ListMigrationTaskssezione AWS SDK for Java 2.x API Reference.

Esempi di Amazon MSK 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 MSK.

Ogni esempio include un collegamento al codice sorgente completo, dove puoi trovare istruzioni su come configurare ed eseguire il codice nel contesto.

Argomenti

- Esempi serverless

Esempi serverless

Invocare una funzione Lambda da un trigger Amazon MSK

Il seguente esempio di codice mostra come implementare una funzione Lambda che riceve un evento generato dalla ricezione di record da un cluster Amazon MSK. La funzione recupera il payload MSK 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 Amazon MSK con Lambda tramite Java.

```
import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.KafkaEvent;
import com.amazonaws.services.lambda.runtime.events.KafkaEvent.KafkaEventRecord;

import java.util.Base64;
import java.util.Map;

public class Example implements RequestHandler<KafkaEvent, Void> {

    @Override
    public Void handleRequest(KafkaEvent event, Context context) {
        for (Map.Entry<String, java.util.List<KafkaEventRecord>> entry :
 event.getRecords().entrySet()) {
            String key = entry.getKey();
            System.out.println("Key: " + key);

            for (KafkaEventRecord record : entry.getValue()) {
```

```
                System.out.println("Record: " + record);

                byte[] value = Base64.getDecoder().decode(record.getValue());
                String message = new String(value);
                System.out.println("Message: " + message);
            }
        }

        return null;
    }
}
```

Esempi di Neptune 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 Neptune.

Le nozioni di base sono esempi di codice che mostrano come eseguire le operazioni essenziali all'interno di un servizio.

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.

Gli scenari sono esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio o combinate con altri Servizi AWS.

Ogni esempio include un collegamento al codice sorgente completo, in cui è possibile trovare istruzioni su come configurare ed eseguire il codice nel contesto.

Nozioni di base

Ciao Amazon Neptune

Il seguente esempio di codice mostra come iniziare a usare Neptune.

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 HelloNeptune {
    public static void main(String[] args) {
        NeptuneAsyncClient neptuneClient = NeptuneAsyncClient.create();
        describeDbCluster(neptuneClient).join(); // This ensures the async code runs
 to completion
    }

    /**
     * Describes the Amazon Neptune DB clusters.
     *
     * @param neptuneClient the Neptune asynchronous client used to make the request
     * @return a {@link CompletableFuture} that completes when the operation is
 finished
     */
    public static CompletableFuture<Void> describeDbCluster(NeptuneAsyncClient
 neptuneClient) {
        DescribeDbClustersRequest request = DescribeDbClustersRequest.builder()
                .maxRecords(20)
                .build();

        SdkPublisher<DescribeDbClustersResponse> paginator =
 neptuneClient.describeDBClustersPaginator(request);
        CompletableFuture<Void> future = new CompletableFuture<>();

        paginator.subscribe(new Subscriber<DescribeDbClustersResponse>() {
            private Subscription subscription;
```

```
            @Override
            public void onSubscribe(Subscription s) {
                this.subscription = s;
                s.request(Long.MAX_VALUE); // request all items
            }

            @Override
            public void onNext(DescribeDbClustersResponse response) {
                response.dbClusters().forEach(cluster -> {
                    System.out.println("Cluster Identifier: " +
cluster.dbClusterIdentifier());
                    System.out.println("Status: " + cluster.status());
                });
            }

            @Override
            public void onError(Throwable t) {
                future.completeExceptionally(t);
            }

            @Override
            public void onComplete() {
                future.complete(null);
            }
        });

        return future.whenComplete((result, throwable) -> {
            neptuneClient.close();
            if (throwable != null) {
                System.err.println("Error describing DB clusters: " +
throwable.getMessage());
            }
        });
    }
```

- Per i dettagli sull'API, consulta Descrivi DBClusters Paginator in AWS SDK for Java 2.x API Reference.

Argomenti

- Nozioni di base

- Azioni
- Scenari

Nozioni di base

Informazioni di base

L'esempio di codice seguente mostra come:

- Crea un sottogruppo Amazon Neptune.
- Crea un ammasso di Nettune.
- Crea un'istanza di Neptune.
- Controlla lo stato dell'istanza di Neptune.
- Mostra i dettagli del cluster Neptune.
- Fermate l'ammasso di Neptune.
- Avvia il cluster Neptune.
- Eliminate le risorse di Neptune.

SDK per Java 2.x

> **Note**
>
> C'è altro da fare. 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 Neptune.

```
public class NeptuneScenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");
    private static final Logger logger =
 LoggerFactory.getLogger(NeptuneScenario.class);
    static Scanner scanner = new Scanner(System.in);
    static NeptuneActions neptuneActions = new NeptuneActions();

    public static void main(String[] args) {
        final String usage =
```

```
                """
                Usage:
                    <subnetGroupName> <clusterName> <dbInstanceId>

                Where:
                    subnetGroupName - The name of an existing Neptune DB subnet
group that includes subnets in at least two Availability Zones.
                    clusterName     - The unique identifier for the Neptune DB
cluster.
                    dbInstanceId    - The identifier for a specific Neptune DB
instance within the cluster.
                """;
        String subnetGroupName = "neptuneSubnetGroup65";
        String clusterName = "neptuneCluster65";
        String dbInstanceId = "neptuneDB65";

        logger.info("""
                   Amazon Neptune is a fully managed graph
                   database service by AWS, designed specifically
                   for handling complex relationships and connected
                   datasets at scale. It supports two popular graph models:
                   property graphs (via openCypher and Gremlin) and RDF
                   graphs (via SPARQL). This makes Neptune ideal for
                   use cases such as knowledge graphs, fraud detection,
                   social networking, recommendation engines, and
                   network management, where relationships between
                   entities are central to the data.

                   Being fully managed, Neptune handles database
                   provisioning, patching, backups, and replication,
                   while also offering high availability and durability
                   within AWS's infrastructure.

                   For developers, programming with Neptune allows
                   for building intelligent, relationship-aware
                   applications that go beyond traditional tabular
                   databases. Developers can use the AWS SDK for Java
                   to automate infrastructure operations (via NeptuneClient).

                    Let's get started...
                    """);
        waitForInputToContinue(scanner);
        runScenario(subnetGroupName, dbInstanceId, clusterName);
    }
```

```
   public static void runScenario(String subnetGroupName, String dbInstanceId,
String clusterName) {
       logger.info(DASHES);
       logger.info("1. Create a Neptune DB Subnet Group");
       logger.info("The Neptune DB subnet group is used when launching a Neptune
cluster");
       waitForInputToContinue(scanner);
       try {
           neptuneActions.createSubnetGroupAsync(subnetGroupName).join();

       } catch (CompletionException ce) {
           Throwable cause = ce.getCause();
           if (cause instanceof ServiceQuotaExceededException) {
               logger.error("The request failed due to service quota exceeded: {}",
cause.getMessage());
           } else {
               logger.error("An unexpected error occurred.", cause);
           }
           return;
       }
       waitForInputToContinue(scanner);
       logger.info(DASHES);

       logger.info(DASHES);
       logger.info("2. Create a Neptune Cluster");
       logger.info("A Neptune Cluster allows you to store and query highly
connected datasets with low latency.");
       waitForInputToContinue(scanner);
       String dbClusterId;
       try {
           dbClusterId = neptuneActions.createDBClusterAsync(clusterName).join();
       } catch (CompletionException ce) {
           Throwable cause = ce.getCause();
           if (cause instanceof ServiceQuotaExceededException) {
               logger.error("The request failed due to service quota exceeded: {}",
cause.getMessage());
           } else {
               logger.error("An unexpected error occurred.", cause);
           }
           return;
       }

       waitForInputToContinue(scanner);
```

```
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("3. Create a Neptune DB Instance");
        logger.info("In this step, we add a new database instance to the Neptune
cluster");
        waitForInputToContinue(scanner);
        try {
        neptuneActions.createDBInstanceAsync(dbInstanceId, dbClusterId).join();
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ServiceQuotaExceededException) {
                logger.error("The request failed due to service quota exceeded: {}",
cause.getMessage());
            } else {
                logger.error("An unexpected error occurred.", cause);
            }
            return;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("4. Check the status of the Neptune DB Instance");
        logger.info("""
                    In this step, we will wait until the DB instance
                    becomes available. This may take around 10 minutes.
                    """);
        waitForInputToContinue(scanner);
        try {
            neptuneActions.checkInstanceStatus(dbInstanceId, "available").join();
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            logger.error("An unexpected error occurred.", cause);
            return;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("5.Show Neptune Cluster details");
        waitForInputToContinue(scanner);
        try {
            neptuneActions.describeDBClustersAsync(clusterName).join();
```

```
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ResourceNotFoundException) {
                logger.error("The request failed due to the resource not found: {}",
cause.getMessage());
            } else {
                logger.error("An unexpected error occurred.", cause);
            }
            return;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("6. Stop the Amazon Neptune cluster");
        logger.info("""
                    Once stopped, this step polls the status
                    until the cluster is in a stopped state.
                    """);
        waitForInputToContinue(scanner);
        try {
            neptuneActions.stopDBClusterAsync(dbClusterId);
            neptuneActions.waitForClusterStatus(dbClusterId, "stopped");
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ResourceNotFoundException) {
                logger.error("The request failed due to the resource not found: {}",
cause.getMessage());
            } else {
                logger.error("An unexpected error occurred.", cause);
            }
            return;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("7. Start the Amazon Neptune cluster");
        logger.info("""
                    Once started, this step polls the clusters
                    status until it's in an available state.
                    We will also poll the instance status.
                    """);
        waitForInputToContinue(scanner);
```

```
        try {
            neptuneActions.startDBClusterAsync(dbClusterId);
            neptuneActions.waitForClusterStatus(dbClusterId, "available");
            neptuneActions.checkInstanceStatus(dbInstanceId, "available").join();
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ResourceNotFoundException) {
                logger.error("The request failed due to the resource not found: {}",
cause.getMessage());
            } else {
                logger.error("An unexpected error occurred.", cause);
            }
            return;
        }
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("8. Delete the Neptune Assets");
        logger.info("Would you like to delete the Neptune Assets? (y/n)");
        String delAns = scanner.nextLine().trim();
        if (delAns.equalsIgnoreCase("y")) {
            logger.info("You selected to delete the Neptune assets.");
            try {
                neptuneActions.deleteNeptuneResourcesAsync(dbInstanceId,
clusterName, subnetGroupName);
            } catch (CompletionException ce) {
                Throwable cause = ce.getCause();
                if (cause instanceof ResourceNotFoundException) {
                    logger.error("The request failed due to the resource not found:
{}", cause.getMessage());
                } else {
                    logger.error("An unexpected error occurred.", cause);
                }
                return;
            }
        } else {
            logger.info("You selected not to delete Neptune assets.");
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info(
                """
```

```
                Thank you for checking out the Amazon Neptune Service Use demo. We
 hope you
                learned something new, or got some inspiration for your own apps
 today.
                For more AWS code examples, have a look at:
                https://docs.aws.amazon.com/code-library/latest/ug/what-is-code-
library.html
                """);
        logger.info(DASHES);
    }

    private static void waitForInputToContinue(Scanner scanner) {
        while (true) {
            logger.info("");
            logger.info("Enter 'c' followed by <ENTER> to continue:");
            String input = scanner.nextLine();

            if (input.trim().equalsIgnoreCase("c")) {
                logger.info("Continuing with the program...");
                logger.info("");
                break;
            } else {
                logger.info("Invalid input. Please try again.");
            }
        }
    }
}
```

Una classe wrapper per i metodi Neptune SDK.

```
public class NeptuneActions {
    private CompletableFuture<Void> instanceCheckFuture;
    private static NeptuneAsyncClient neptuneAsyncClient;
    private final Region region = Region.US_EAST_1;
    private static final Logger logger =
 LoggerFactory.getLogger(NeptuneActions.class);
    private final NeptuneClient neptuneClient =
 NeptuneClient.builder().region(region).build();

    /**
     * Retrieves an instance of the NeptuneAsyncClient.
     * <p>
```

```
    * This method initializes and returns a singleton instance of the
NeptuneAsyncClient. The client
    * is configured with the following settings:
    * <ul>
    *     <li>Maximum concurrency: 100</li>
    *     <li>Connection timeout: 60 seconds</li>
    *     <li>Read timeout: 60 seconds</li>
    *     <li>Write timeout: 60 seconds</li>
    *     <li>API call timeout: 2 minutes</li>
    *     <li>API call attempt timeout: 90 seconds</li>
    *     <li>Retry strategy: STANDARD</li>
    * </ul>
    * The client is built using the NettyNioAsyncHttpClient.
    *
    * @return the singleton instance of the NeptuneAsyncClient
    */
   private static NeptuneAsyncClient getAsyncClient() {
       if (neptuneAsyncClient == null) {
           SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                   .maxConcurrency(100)
                   .connectionTimeout(Duration.ofSeconds(60))
                   .readTimeout(Duration.ofSeconds(60))
                   .writeTimeout(Duration.ofSeconds(60))
                   .build();

           ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
                   .apiCallTimeout(Duration.ofMinutes(2))
                   .apiCallAttemptTimeout(Duration.ofSeconds(90))
                   .retryStrategy(RetryMode.STANDARD)
                   .build();

           neptuneAsyncClient = NeptuneAsyncClient.builder()
                   .httpClient(httpClient)
                   .overrideConfiguration(overrideConfig)
                   .build();
       }
       return neptuneAsyncClient;
   }

   /**
    * Asynchronously deletes a set of Amazon Neptune resources in a defined order.
    * <p>
    * The method performs the following operations in sequence:
```

```
     * <ol>
     *     <li>Deletes the Neptune DB instance identified by {@code dbInstanceId}.</
li>
     *     <li>Waits until the DB instance is fully deleted.</li>
     *     <li>Deletes the Neptune DB cluster identified by {@code dbClusterId}.</
li>
     *     <li>Deletes the Neptune DB subnet group identified by {@code
 subnetGroupName}.</li>
     * </ol>
     * <p>
     * If any step fails, the subsequent operations are not performed, and the
 exception
     * is logged. This method blocks the calling thread until all operations
 complete.
     *
     * @param dbInstanceId      the ID of the Neptune DB instance to delete
     * @param dbClusterId       the ID of the Neptune DB cluster to delete
     * @param subnetGroupName   the name of the Neptune DB subnet group to delete
     */
    public void deleteNeptuneResourcesAsync(String dbInstanceId, String dbClusterId,
 String subnetGroupName) {
        deleteDBInstanceAsync(dbInstanceId)
                .thenCompose(v -> waitUntilInstanceDeletedAsync(dbInstanceId))
                .thenCompose(v -> deleteDBClusterAsync(dbClusterId))
                .thenCompose(v -> deleteDBSubnetGroupAsync(subnetGroupName))
                .whenComplete((v, ex) -> {
                    if (ex != null) {
                        logger.info("Failed to delete Neptune resources: " +
 ex.getMessage());
                    } else {
                        logger.info("Neptune resources deleted successfully.");
                    }
                })
                .join(); // Waits for the entire async chain to complete
    }

    /**
     * Deletes a subnet group.
     *
     * @param subnetGroupName the identifier of the subnet group to delete
     * @return a {@link CompletableFuture} that completes when the cluster has been
 deleted
     */
```

```
   public CompletableFuture<Void> deleteDBSubnetGroupAsync(String subnetGroupName)
{
       DeleteDbSubnetGroupRequest request = DeleteDbSubnetGroupRequest.builder()
               .dbSubnetGroupName(subnetGroupName)
               .build();

       return getAsyncClient().deleteDBSubnetGroup(request)
               .thenAccept(response -> logger.info("## Deleting Subnet Group: " +
subnetGroupName));
   }

   /**
    * Deletes a DB instance asynchronously.
    *
    * @param clusterId the identifier of the cluster to delete
    * @return a {@link CompletableFuture} that completes when the cluster has been
deleted
    */
   public CompletableFuture<Void> deleteDBClusterAsync(String clusterId) {
       DeleteDbClusterRequest request = DeleteDbClusterRequest.builder()
               .dbClusterIdentifier(clusterId)
               .skipFinalSnapshot(true)
               .build();

       return getAsyncClient().deleteDBCluster(request)
               .thenAccept(response -> System.out.println("## Deleting DB Cluster:
" + clusterId));
   }

   public CompletableFuture<Void> waitUntilInstanceDeletedAsync(String instanceId)
{
       CompletableFuture<Void> future = new CompletableFuture<>();
       long startTime = System.currentTimeMillis();
       checkInstanceDeletedRecursive(instanceId, startTime, future);
       return future;
   }

   /**
    * Deletes a DB instance asynchronously.
    *
    * @param instanceId the identifier of the DB instance to be deleted
    * @return a {@link CompletableFuture} that completes when the DB instance has
been deleted
    */
```

```
   public CompletableFuture<Void> deleteDBInstanceAsync(String instanceId) {
       DeleteDbInstanceRequest request = DeleteDbInstanceRequest.builder()
               .dbInstanceIdentifier(instanceId)
               .skipFinalSnapshot(true)
               .build();

       return getAsyncClient().deleteDBInstance(request)
               .thenAccept(response -> System.out.println("## Deleting DB Instance:
" + instanceId));
   }


   private void checkInstanceDeletedRecursive(String instanceId, long startTime,
CompletableFuture<Void> future) {
       DescribeDbInstancesRequest request = DescribeDbInstancesRequest.builder()
               .dbInstanceIdentifier(instanceId)
               .build();

       getAsyncClient().describeDBInstances(request)
               .whenComplete((response, exception) -> {
                   if (exception != null) {
                       Throwable cause = exception.getCause();
                       if (cause instanceof NeptuneException &&
                               ((NeptuneException)
cause).awsErrorDetails().errorCode().equals("DBInstanceNotFound")) {
                           long elapsed = (System.currentTimeMillis() -
startTime) / 1000;
                           logger.info("\r Instance %s deleted after %ds%n",
instanceId, elapsed);
                           future.complete(null);
                           return;
                       }
                       future.completeExceptionally(new CompletionException("Error
polling DB instance", cause));
                       return;
                   }

                   String status =
response.dbInstances().get(0).dbInstanceStatus();
                   long elapsed = (System.currentTimeMillis() - startTime) / 1000;
                   System.out.printf("\r  Waiting: Instance %s status: %-10s (%ds
elapsed)", instanceId, status, elapsed);
                   System.out.flush();
```

```
                    CompletableFuture.delayedExecutor(20, TimeUnit.SECONDS)
                            .execute(() -> checkInstanceDeletedRecursive(instanceId,
startTime, future));
                });
    }


    public void waitForClusterStatus(String clusterId, String desiredStatus) {
        System.out.printf("Waiting for cluster '%s' to reach status '%s'...\n",
clusterId, desiredStatus);
        CompletableFuture<Void> future = new CompletableFuture<>();
        checkClusterStatusRecursive(clusterId, desiredStatus,
System.currentTimeMillis(), future);
        future.join();
    }

    private void checkClusterStatusRecursive(String clusterId, String desiredStatus,
long startTime, CompletableFuture<Void> future) {
        DescribeDbClustersRequest request = DescribeDbClustersRequest.builder()
                .dbClusterIdentifier(clusterId)
                .build();

        getAsyncClient().describeDBClusters(request)
                .whenComplete((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause();
                        future.completeExceptionally(
                                new CompletionException("Error checking Neptune
cluster status", cause)
                        );
                        return;
                    }

                    List<DBCluster> clusters = response.dbClusters();
                    if (clusters.isEmpty()) {
                        future.completeExceptionally(new RuntimeException("Cluster
not found: " + clusterId));
                        return;
                    }

                    String currentStatus = clusters.get(0).status();
                    long elapsedSeconds = (System.currentTimeMillis() - startTime) /
1000;
```

```
                    System.out.printf("\r Elapsed: %-20s  Cluster status: %-20s",
formatElapsedTime((int) elapsedSeconds), currentStatus);
                    System.out.flush();

                    if (desiredStatus.equalsIgnoreCase(currentStatus)) {
                        System.out.printf("\r Neptune cluster reached desired status
'%s' after %s.\n", desiredStatus, formatElapsedTime((int) elapsedSeconds));
                        future.complete(null);
                    } else {
                        CompletableFuture.delayedExecutor(20, TimeUnit.SECONDS)
                                .execute(() ->
checkClusterStatusRecursive(clusterId, desiredStatus, startTime, future));
                    }
                });
    }


    /**
     * Starts an Amazon Neptune DB cluster.
     *
     * @param clusterIdentifier the unique identifier of the DB cluster to be
stopped
     */
    public CompletableFuture<StartDbClusterResponse> startDBClusterAsync(String
clusterIdentifier) {
        StartDbClusterRequest clusterRequest = StartDbClusterRequest.builder()
                .dbClusterIdentifier(clusterIdentifier)
                .build();

        return getAsyncClient().startDBCluster(clusterRequest)
                .whenComplete((response, error) -> {
                    if (error != null) {
                        Throwable cause = error.getCause() != null ?
error.getCause() : error;

                        if (cause instanceof ResourceNotFoundException) {
                            throw (ResourceNotFoundException) cause;
                        }

                        throw new RuntimeException("Failed to start DB cluster: " +
cause.getMessage(), cause);
                    } else {
                        logger.info("DB Cluster starting: " + clusterIdentifier);
                    }
```

```
                });
    }

    /**
     * Stops an Amazon Neptune DB cluster.
     *
     * @param clusterIdentifier the unique identifier of the DB cluster to be
stopped
     */
    public CompletableFuture<StopDbClusterResponse> stopDBClusterAsync(String
clusterIdentifier) {
        StopDbClusterRequest clusterRequest = StopDbClusterRequest.builder()
                .dbClusterIdentifier(clusterIdentifier)
                .build();

        return getAsyncClient().stopDBCluster(clusterRequest)
                .whenComplete((response, error) -> {
                    if (error != null) {
                        Throwable cause = error.getCause() != null ?
error.getCause() : error;

                        if (cause instanceof ResourceNotFoundException) {
                            throw (ResourceNotFoundException) cause;
                        }

                        throw new RuntimeException("Failed to stop DB cluster: " +
cause.getMessage(), cause);
                    } else {
                        logger.info("DB Cluster stopped: " + clusterIdentifier);
                    }
                });
    }



    /**
     * Asynchronously describes the specified Amazon RDS DB cluster.
     *
     * @param clusterId the identifier of the DB cluster to describe
     * @return a {@link CompletableFuture} that completes when the operation is
done, or throws a {@link RuntimeException}
     * if an error occurs
     */
    public CompletableFuture<Void> describeDBClustersAsync(String clusterId) {
```

```
        DescribeDbClustersRequest request = DescribeDbClustersRequest.builder()
                .dbClusterIdentifier(clusterId)
                .build();

        return getAsyncClient().describeDBClusters(request)
                .thenAccept(response -> {
                    for (DBCluster cluster : response.dbClusters()) {
                        logger.info("Cluster Identifier: " +
cluster.dbClusterIdentifier());
                        logger.info("Status: " + cluster.status());
                        logger.info("Engine: " + cluster.engine());
                        logger.info("Engine Version: " + cluster.engineVersion());
                        logger.info("Endpoint: " + cluster.endpoint());
                        logger.info("Reader Endpoint: " + cluster.readerEndpoint());
                        logger.info("Availability Zones: " +
cluster.availabilityZones());
                        logger.info("Subnet Group: " + cluster.dbSubnetGroup());
                        logger.info("VPC Security Groups:");
                        cluster.vpcSecurityGroups().forEach(vpcGroup ->
                                logger.info("  - " +
vpcGroup.vpcSecurityGroupId()));
                        logger.info("Storage Encrypted: " +
cluster.storageEncrypted());
                        logger.info("IAM DB Auth Enabled: " +
cluster.iamDatabaseAuthenticationEnabled());
                        logger.info("Backup Retention Period: " +
cluster.backupRetentionPeriod() + " days");
                        logger.info("Preferred Backup Window: " +
cluster.preferredBackupWindow());
                        logger.info("Preferred Maintenance Window: " +
cluster.preferredMaintenanceWindow());
                        logger.info("------");
                    }
                })
                .exceptionally(ex -> {
                    Throwable cause = ex.getCause() != null ? ex.getCause() : ex;

                    if (cause instanceof ResourceNotFoundException) {
                        throw (ResourceNotFoundException) cause;
                    }

                    throw new RuntimeException("Failed to describe the DB cluster: "
+ cause.getMessage(), cause);
                });
```

```
    }


    public CompletableFuture<Void> checkInstanceStatus(String instanceId, String
desiredStatus) {
        CompletableFuture<Void> future = new CompletableFuture<>();
        long startTime = System.currentTimeMillis();
        checkStatusRecursive(instanceId, desiredStatus.toLowerCase(), startTime,
future);
        return future;
    }

    /**
     * Checks the status of a Neptune instance recursively until the desired status
is reached or a timeout occurs.
     *
     * @param instanceId     the ID of the Neptune instance to check
     * @param desiredStatus  the desired status of the Neptune instance
     * @param startTime      the start time of the operation, used to calculate the
elapsed time
     * @param future         a {@link CompletableFuture} that will be completed when
the desired status is reached
     */
    private void checkStatusRecursive(String instanceId, String desiredStatus, long
startTime, CompletableFuture<Void> future) {
        DescribeDbInstancesRequest request = DescribeDbInstancesRequest.builder()
                .dbInstanceIdentifier(instanceId)
                .build();

        getAsyncClient().describeDBInstances(request)
                .whenComplete((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause();
                        future.completeExceptionally(
                                new CompletionException("Error checking Neptune
instance status", cause)
                        );
                        return;
                    }

                    List<DBInstance> instances = response.dbInstances();
                    if (instances.isEmpty()) {
                        future.completeExceptionally(new RuntimeException("Instance
not found: " + instanceId));
```

```
                        return;
                    }

                    String currentStatus = instances.get(0).dbInstanceStatus();
                    long elapsedSeconds = (System.currentTimeMillis() - startTime) /
1000;
                    System.out.printf("\r Elapsed: %-20s  Status: %-20s",
formatElapsedTime((int) elapsedSeconds), currentStatus);
                    System.out.flush();

                    if (desiredStatus.equalsIgnoreCase(currentStatus)) {
                        System.out.printf("\r Neptune instance reached desired
status '%s' after %s.\n", desiredStatus, formatElapsedTime((int) elapsedSeconds));
                        future.complete(null);
                    } else {
                        CompletableFuture.delayedExecutor(20, TimeUnit.SECONDS)
                                .execute(() -> checkStatusRecursive(instanceId,
desiredStatus, startTime, future));
                    }
                });
    }


    private String formatElapsedTime(int seconds) {
        int minutes = seconds / 60;
        int remainingSeconds = seconds % 60;

        if (minutes > 0) {
            return minutes + (minutes == 1 ? " min" : " mins") + ", " +
                    remainingSeconds + (remainingSeconds == 1 ? " sec" : " secs");
        } else {
            return remainingSeconds + (remainingSeconds == 1 ? " sec" : " secs");
        }
    }


    /**
     * Creates a new Amazon Neptune DB instance asynchronously.
     *
     * @param dbInstanceId the identifier for the new DB instance
     * @param dbClusterId  the identifier for the DB cluster that the new instance
will be a part of
     * @return a {@link CompletableFuture} that completes with the identifier of the
newly created DB instance
```

```
    * @throws CompletionException if the operation fails, with a cause of either:
    *                             - {@link ServiceQuotaExceededException} if the
request would exceed the maximum quota, or
    *                             - a general exception with the failure message
    */
   public CompletableFuture<String> createDBInstanceAsync(String dbInstanceId,
String dbClusterId) {
       CreateDbInstanceRequest request = CreateDbInstanceRequest.builder()
               .dbInstanceIdentifier(dbInstanceId)
               .dbInstanceClass("db.r5.large")
               .engine("neptune")
               .dbClusterIdentifier(dbClusterId)
               .build();

       return getAsyncClient().createDBInstance(request)
               .whenComplete((response, exception) -> {
                   if (exception != null) {
                       Throwable cause = exception.getCause();
                       if (cause instanceof ServiceQuotaExceededException) {
                           throw new CompletionException("The operation was denied
because the request would exceed the maximum quota.", cause);
                       }
                       throw new CompletionException("Failed to create Neptune DB
instance: " + exception.getMessage(), exception);
                   }
               })
               .thenApply(response -> {
                   String instanceId =
response.dbInstance().dbInstanceIdentifier();
                   logger.info("Created Neptune DB Instance: " + instanceId);
                   return instanceId;
               });
   }


   /**
    * Creates a new Amazon Neptune DB cluster asynchronously.
    *
    * @param dbName the name of the DB cluster to be created
    * @return a CompletableFuture that, when completed, provides the ID of the
created DB cluster
    * @throws CompletionException if the operation fails for any reason, including
if the request would exceed the maximum quota
    */
```

```
   public CompletableFuture<String> createDBClusterAsync(String dbName) {
       CreateDbClusterRequest request = CreateDbClusterRequest.builder()
               .dbClusterIdentifier(dbName)
               .engine("neptune")
               .deletionProtection(false)
               .backupRetentionPeriod(1)
               .build();

       return getAsyncClient().createDBCluster(request)
               .whenComplete((response, exception) -> {
                   if (exception != null) {
                       Throwable cause = exception.getCause();
                       if (cause instanceof ServiceQuotaExceededException) {
                           throw new CompletionException("The operation was denied
because the request would exceed the maximum quota.", cause);
                       }
                       throw new CompletionException("Failed to create Neptune DB
cluster: " + exception.getMessage(), exception);
                   }
               })
               .thenApply(response -> {
                   String clusterId = response.dbCluster().dbClusterIdentifier();
                   logger.info("DB Cluster created: " + clusterId);
                   return clusterId;
               });
   }


   /**
    * Creates a new DB subnet group asynchronously.
    *
    * @param groupName the name of the subnet group to create
    * @return a CompletableFuture that, when completed, returns the Amazon Resource
Name (ARN) of the created subnet group
    * @throws CompletionException if the operation fails, with a cause that may be
a ServiceQuotaExceededException if the request would exceed the maximum quota
    */
   public CompletableFuture<String> createSubnetGroupAsync(String groupName) {

       // Get the Amazon Virtual Private Cloud (VPC) where the Neptune cluster and
resources will be created
       String vpcId = getDefaultVpcId();
       logger.info("VPC is : " + vpcId);
```

```
        List<String> subnetList = getSubnetIds(vpcId);
        for (String subnetId : subnetList) {
            System.out.println("Subnet group:" +subnetId);
        }

        CreateDbSubnetGroupRequest request = CreateDbSubnetGroupRequest.builder()
                .dbSubnetGroupName(groupName)
                .dbSubnetGroupDescription("Subnet group for Neptune cluster")
                .subnetIds(subnetList)
                .build();

        return getAsyncClient().createDBSubnetGroup(request)
                .whenComplete((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause();
                        if (cause instanceof ServiceQuotaExceededException) {
                            throw new CompletionException("The operation was denied
because the request would exceed the maximum quota.", cause);
                        }
                        throw new CompletionException("Failed to create subnet
group: " + exception.getMessage(), exception);
                    }
                })
                .thenApply(response -> {
                    String name = response.dbSubnetGroup().dbSubnetGroupName();
                    String arn = response.dbSubnetGroup().dbSubnetGroupArn();
                    logger.info("Subnet group created: " + name);
                    return arn;
                });
    }

    private List<String> getSubnetIds(String vpcId) {
        try (Ec2Client ec2 = Ec2Client.builder().region(region).build()) {
            DescribeSubnetsRequest request = DescribeSubnetsRequest.builder()
                    .filters(builder -> builder.name("vpc-id").values(vpcId))
                    .build();

            DescribeSubnetsResponse response = ec2.describeSubnets(request);
            return response.subnets().stream()
                    .map(Subnet::subnetId)
                    .collect(Collectors.toList());
        }
    }
```

```
    public static String getDefaultVpcId() {
        Ec2Client ec2 = Ec2Client.builder()
                .region(Region.US_EAST_1)
                .build();

        Filter myFilter = Filter.builder()
                .name("isDefault")
                .values("true")
                .build();

        List<Filter> filterList = new ArrayList<>();
        filterList.add(myFilter);

        DescribeVpcsRequest request = DescribeVpcsRequest.builder()
                .filters(filterList)
                .build();


        DescribeVpcsResponse response = ec2.describeVpcs(request);
        if (!response.vpcs().isEmpty()) {
            Vpc defaultVpc = response.vpcs().get(0);
            return defaultVpc.vpcId();
        } else {
            throw new RuntimeException("No default VPC found in this region.");
        }
    }
}
```

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.

```
   /**
    * Creates a new Amazon Neptune DB cluster asynchronously.
    *
    * @param dbName the name of the DB cluster to be created
    * @return a CompletableFuture that, when completed, provides the ID of the
created DB cluster
    * @throws CompletionException if the operation fails for any reason, including
if the request would exceed the maximum quota
    */
   public CompletableFuture<String> createDBClusterAsync(String dbName) {
       CreateDbClusterRequest request = CreateDbClusterRequest.builder()
               .dbClusterIdentifier(dbName)
               .engine("neptune")
               .deletionProtection(false)
               .backupRetentionPeriod(1)
               .build();

       return getAsyncClient().createDBCluster(request)
               .whenComplete((response, exception) -> {
                   if (exception != null) {
                       Throwable cause = exception.getCause();
                       if (cause instanceof ServiceQuotaExceededException) {
                           throw new CompletionException("The operation was denied
because the request would exceed the maximum quota.", cause);
                       }
                       throw new CompletionException("Failed to create Neptune DB
cluster: " + exception.getMessage(), exception);
                   }
               })
               .thenApply(response -> {
                   String clusterId = response.dbCluster().dbClusterIdentifier();
                   logger.info("DB Cluster created: " + clusterId);
                   return clusterId;
               });
   }
```

- Per i dettagli sull'API, consulta Create DBCluster 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.

```
   /**
    * Creates a new Amazon Neptune DB instance asynchronously.
    *
    * @param dbInstanceId the identifier for the new DB instance
    * @param dbClusterId  the identifier for the DB cluster that the new instance
will be a part of
    * @return a {@link CompletableFuture} that completes with the identifier of the
newly created DB instance
    * @throws CompletionException if the operation fails, with a cause of either:
    *                             - {@link ServiceQuotaExceededException} if the
request would exceed the maximum quota, or
    *                             - a general exception with the failure message
    */
   public CompletableFuture<String> createDBInstanceAsync(String dbInstanceId,
String dbClusterId) {
       CreateDbInstanceRequest request = CreateDbInstanceRequest.builder()
               .dbInstanceIdentifier(dbInstanceId)
               .dbInstanceClass("db.r5.large")
               .engine("neptune")
               .dbClusterIdentifier(dbClusterId)
               .build();

       return getAsyncClient().createDBInstance(request)
               .whenComplete((response, exception) -> {
                   if (exception != null) {
                       Throwable cause = exception.getCause();
                       if (cause instanceof ServiceQuotaExceededException) {
                           throw new CompletionException("The operation was denied
because the request would exceed the maximum quota.", cause);
                       }
```

```
                        throw new CompletionException("Failed to create Neptune DB
instance: " + exception.getMessage(), exception);
                    }
                })
                .thenApply(response -> {
                    String instanceId =
response.dbInstance().dbInstanceIdentifier();
                    logger.info("Created Neptune DB Instance: " + instanceId);
                    return instanceId;
                });
    }
```

- Per i dettagli sull'API, consulta Create DBInstance in AWS SDK for Java 2.x API Reference.

`CreateDBSubnetGroup`

Il seguente esempio di codice mostra come utilizzare`CreateDBSubnetGroup`.

SDK per Java 2.x

> **Note**
>
> C'è 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 a new DB subnet group asynchronously.
     *
     * @param groupName the name of the subnet group to create
     * @return a CompletableFuture that, when completed, returns the Amazon Resource
Name (ARN) of the created subnet group
     * @throws CompletionException if the operation fails, with a cause that may be
a ServiceQuotaExceededException if the request would exceed the maximum quota
     */
    public CompletableFuture<String> createSubnetGroupAsync(String groupName) {

        // Get the Amazon Virtual Private Cloud (VPC) where the Neptune cluster and
resources will be created
```

```
        String vpcId = getDefaultVpcId();
        logger.info("VPC is : " + vpcId);

        List<String> subnetList = getSubnetIds(vpcId);
        for (String subnetId : subnetList) {
            System.out.println("Subnet group:" +subnetId);
        }

        CreateDbSubnetGroupRequest request = CreateDbSubnetGroupRequest.builder()
                .dbSubnetGroupName(groupName)
                .dbSubnetGroupDescription("Subnet group for Neptune cluster")
                .subnetIds(subnetList)
                .build();

        return getAsyncClient().createDBSubnetGroup(request)
                .whenComplete((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause();
                        if (cause instanceof ServiceQuotaExceededException) {
                            throw new CompletionException("The operation was denied
because the request would exceed the maximum quota.", cause);
                        }
                        throw new CompletionException("Failed to create subnet
group: " + exception.getMessage(), exception);
                    }
                })
                .thenApply(response -> {
                    String name = response.dbSubnetGroup().dbSubnetGroupName();
                    String arn = response.dbSubnetGroup().dbSubnetGroupArn();
                    logger.info("Subnet group created: " + name);
                    return arn;
                });
    }
```

- Per i dettagli sull'API, consulta Create DBSubnet Group in AWS SDK for Java 2.x API Reference.

`CreateGraph`

Il seguente esempio di codice mostra come utilizzare`CreateGraph`.

SDK per Java 2.x

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
   /**
    * Executes the process of creating a new Neptune graph.
    *
    * @param client        the Neptune graph client used to interact with the
Neptune service
    * @param graphName     the name of the graph to be created
    * @throws NeptuneGraphException if an error occurs while creating the graph
    */
   public static void executeCreateGraph(NeptuneGraphClient client, String
graphName) {
       try {
           // Create the graph request
           CreateGraphRequest request = CreateGraphRequest.builder()
                   .graphName(graphName)
                   .provisionedMemory(16)
                   .build();

           // Create the graph
           CreateGraphResponse response = client.createGraph(request);

           // Extract the graph name and ARN
           String createdGraphName = response.name();
           String graphArn = response.arn();
           String graphEndpoint = response.endpoint();

           System.out.println("Graph created successfully!");
           System.out.println("Graph Name: " + createdGraphName);
           System.out.println("Graph ARN: " + graphArn);
           System.out.println("Graph Endpoint: " +graphEndpoint );

       } catch (NeptuneGraphException e) {
           System.err.println("Failed to create graph: " +
e.awsErrorDetails().errorMessage());
       } finally {
```

```
            client.close();
        }
    }
```

- Per i dettagli sull'API, consulta la CreateGraphsezione AWS SDK for Java 2.x API Reference.

`DeleteDBCluster`

Il seguente esempio di codice mostra come utilizzare`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.

```
    /**
     * Deletes a DB instance asynchronously.
     *
     * @param clusterId the identifier of the cluster to delete
     * @return a {@link CompletableFuture} that completes when the cluster has been
deleted
     */
    public CompletableFuture<Void> deleteDBClusterAsync(String clusterId) {
        DeleteDbClusterRequest request = DeleteDbClusterRequest.builder()
                .dbClusterIdentifier(clusterId)
                .skipFinalSnapshot(true)
                .build();

        return getAsyncClient().deleteDBCluster(request)
                .thenAccept(response -> System.out.println("## Deleting DB Cluster:
" + clusterId));
    }
```

- Per i dettagli sull'API, consulta Delete DBCluster 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.

```
   /**
    * Deletes a DB instance asynchronously.
    *
    * @param instanceId the identifier of the DB instance to be deleted
    * @return a {@link CompletableFuture} that completes when the DB instance has
been deleted
    */
   public CompletableFuture<Void> deleteDBInstanceAsync(String instanceId) {
       DeleteDbInstanceRequest request = DeleteDbInstanceRequest.builder()
               .dbInstanceIdentifier(instanceId)
               .skipFinalSnapshot(true)
               .build();

       return getAsyncClient().deleteDBInstance(request)
               .thenAccept(response -> System.out.println("## Deleting DB Instance:
" + instanceId));
   }
```

- Per i dettagli sull'API, consulta Delete DBInstance in AWS SDK for Java 2.x API Reference.

`DeleteDBSubnetGroup`

Il seguente esempio di codice mostra come utilizzare`DeleteDBSubnetGroup`.

SDK per Java 2.x

> **Note**
>
> C'è 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 subnet group.
     *
     * @param subnetGroupName the identifier of the subnet group to delete
     * @return a {@link CompletableFuture} that completes when the cluster has been
deleted
     */
    public CompletableFuture<Void> deleteDBSubnetGroupAsync(String subnetGroupName)
{
        DeleteDbSubnetGroupRequest request = DeleteDbSubnetGroupRequest.builder()
                .dbSubnetGroupName(subnetGroupName)
                .build();

        return getAsyncClient().deleteDBSubnetGroup(request)
                .thenAccept(response -> logger.info("## Deleting Subnet Group: " +
subnetGroupName));
    }
```

- Per i dettagli sull'API, consulta Delete DBSubnet Group 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.

```
   /**
    * Asynchronously describes the specified Amazon RDS DB cluster.
    *
    * @param clusterId the identifier of the DB cluster to describe
    * @return a {@link CompletableFuture} that completes when the operation is
done, or throws a {@link RuntimeException}
    * if an error occurs
    */
   public CompletableFuture<Void> describeDBClustersAsync(String clusterId) {
       DescribeDbClustersRequest request = DescribeDbClustersRequest.builder()
               .dbClusterIdentifier(clusterId)
               .build();

       return getAsyncClient().describeDBClusters(request)
               .thenAccept(response -> {
                   for (DBCluster cluster : response.dbClusters()) {
                       logger.info("Cluster Identifier: " +
cluster.dbClusterIdentifier());
                       logger.info("Status: " + cluster.status());
                       logger.info("Engine: " + cluster.engine());
                       logger.info("Engine Version: " + cluster.engineVersion());
                       logger.info("Endpoint: " + cluster.endpoint());
                       logger.info("Reader Endpoint: " + cluster.readerEndpoint());
                       logger.info("Availability Zones: " +
cluster.availabilityZones());
                       logger.info("Subnet Group: " + cluster.dbSubnetGroup());
                       logger.info("VPC Security Groups:");
                       cluster.vpcSecurityGroups().forEach(vpcGroup ->
                               logger.info("  - " +
vpcGroup.vpcSecurityGroupId()));
                       logger.info("Storage Encrypted: " +
cluster.storageEncrypted());
                       logger.info("IAM DB Auth Enabled: " +
cluster.iamDatabaseAuthenticationEnabled());
                       logger.info("Backup Retention Period: " +
cluster.backupRetentionPeriod() + " days");
                       logger.info("Preferred Backup Window: " +
cluster.preferredBackupWindow());
                       logger.info("Preferred Maintenance Window: " +
cluster.preferredMaintenanceWindow());
                       logger.info("------");
                   }
```

```
                })
                .exceptionally(ex -> {
                    Throwable cause = ex.getCause() != null ? ex.getCause() : ex;

                    if (cause instanceof ResourceNotFoundException) {
                        throw (ResourceNotFoundException) cause;
                    }

                    throw new RuntimeException("Failed to describe the DB cluster: "
+ cause.getMessage(), cause);
                });
    }
```

- Per i dettagli sull'API, consulta Descrivi DBClusters 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.

```
    /**
     * Checks the status of a Neptune instance recursively until the desired status
is reached or a timeout occurs.
     *
     * @param instanceId     the ID of the Neptune instance to check
     * @param desiredStatus  the desired status of the Neptune instance
     * @param startTime      the start time of the operation, used to calculate the
elapsed time
     * @param future         a {@link CompletableFuture} that will be completed when
the desired status is reached
     */
    private void checkStatusRecursive(String instanceId, String desiredStatus, long
startTime, CompletableFuture<Void> future) {
```

```
        DescribeDbInstancesRequest request = DescribeDbInstancesRequest.builder()
                .dbInstanceIdentifier(instanceId)
                .build();

        getAsyncClient().describeDBInstances(request)
                .whenComplete((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause();
                        future.completeExceptionally(
                                new CompletionException("Error checking Neptune
instance status", cause)
                        );
                        return;
                    }

                    List<DBInstance> instances = response.dbInstances();
                    if (instances.isEmpty()) {
                        future.completeExceptionally(new RuntimeException("Instance
not found: " + instanceId));
                        return;
                    }

                    String currentStatus = instances.get(0).dbInstanceStatus();
                    long elapsedSeconds = (System.currentTimeMillis() - startTime) /
1000;
                    System.out.printf("\r Elapsed: %-20s  Status: %-20s",
formatElapsedTime((int) elapsedSeconds), currentStatus);
                    System.out.flush();

                    if (desiredStatus.equalsIgnoreCase(currentStatus)) {
                        System.out.printf("\r Neptune instance reached desired
status '%s' after %s.\n", desiredStatus, formatElapsedTime((int) elapsedSeconds));
                        future.complete(null);
                    } else {
                        CompletableFuture.delayedExecutor(20, TimeUnit.SECONDS)
                                .execute(() -> checkStatusRecursive(instanceId,
desiredStatus, startTime, future));
                    }
                });
    }
```

- Per i dettagli sull'API, consulta Descrivi DBInstances in AWS SDK for Java 2.x API Reference.

`ExecuteGremlinProfileQuery`

Il seguente esempio di codice mostra come utilizzare`ExecuteGremlinProfileQuery`.

SDK per Java 2.x

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
   /**
    * Executes a Gremlin query against an Amazon Neptune database using the
provided {@link NeptunedataClient}.
    *
    * @param client the {@link NeptunedataClient} instance to use for executing the
Gremlin query
    */
   public static void executeGremlinQuery(NeptunedataClient client) {
       try {
           System.out.println("Querying Neptune...");
           ExecuteGremlinQueryRequest request =
ExecuteGremlinQueryRequest.builder()
                   .gremlinQuery("g.V().has('code', 'ANC')")
                   .build();

           ExecuteGremlinQueryResponse response =
client.executeGremlinQuery(request);

           System.out.println("Full Response:");
           System.out.println(response);

           // Retrieve and print the result
           if (response.result() != null) {
               System.out.println("Query Result:");
               System.out.println(response.result().toString());
           } else {
               System.out.println("No result returned from the query.");
           }
       } catch (NeptunedataException e) {
           System.err.println("Error calling Neptune: " +
e.awsErrorDetails().errorMessage());
```

```
        } catch (Exception e) {
            System.err.println("Unexpected error: " + e.getMessage());
        } finally {
            client.close();
        }
    }
```

- Per i dettagli sull'API, consulta la ExecuteGremlinProfileQuerysezione AWS SDK for Java 2.x API Reference.

`ExecuteGremlinQuery`

Il seguente esempio di codice mostra come utilizzare`ExecuteGremlinQuery`.

SDK per Java 2.x

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
    /**
     * Executes a Gremlin PROFILE query using the provided NeptunedataClient.
     *
     * @param client The NeptunedataClient instance to be used for executing the
Gremlin PROFILE query.
     */
    private static void executeGremlinProfileQuery(NeptunedataClient client) {
        System.out.println("Executing Gremlin PROFILE query...");

        ExecuteGremlinProfileQueryRequest request =
ExecuteGremlinProfileQueryRequest.builder()
                .gremlinQuery("g.V().has('code', 'ANC')")
                .build();

        ExecuteGremlinProfileQueryResponse response =
client.executeGremlinProfileQuery(request);
        if (response.output() != null) {
            System.out.println("Query Profile Output:");
```

```
            System.out.println(response.output());
        } else {
            System.out.println("No output returned from the profile query.");
        }
    }
```

- Per i dettagli sull'API, consulta la ExecuteGremlinQuerysezione AWS SDK for Java 2.x API Reference.

`ExecuteOpenCypherExplainQuery`

Il seguente esempio di codice mostra come utilizzare`ExecuteOpenCypherExplainQuery`.

SDK per Java 2.x

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
    /**
     * Executes an OpenCypher EXPLAIN query using the provided Neptune data client.
     *
     * @param client The Neptune data client to use for the query execution.
     */
    public static void executeGremlinQuery(NeptunedataClient client) {
        try {
            System.out.println("Executing OpenCypher EXPLAIN query...");
            ExecuteOpenCypherExplainQueryRequest request =
ExecuteOpenCypherExplainQueryRequest.builder()
                    .openCypherQuery("MATCH (n {code: 'ANC'}) RETURN n")
                    .explainMode("debug")
                    .build();

            ExecuteOpenCypherExplainQueryResponse response =
client.executeOpenCypherExplainQuery(request);

            if (response.results() != null) {
                System.out.println("Explain Results:");
```

```
                System.out.println(response.results().asUtf8String());
            } else {
                System.out.println("No explain results returned.");
            }

        } catch (NeptunedataException e) {
            System.err.println("Neptune error: " +
e.awsErrorDetails().errorMessage());
        } catch (Exception e) {
            System.err.println("Unexpected error: " + e.getMessage());
        } finally {
            client.close();
        }
    }
```

- Per i dettagli sull'API, consulta la ExecuteOpenCypherExplainQuerysezione AWS SDK for Java 2.x API Reference.

`ExecuteQuery`

Il seguente esempio di codice mostra come utilizzare`ExecuteQuery`.

SDK per Java 2.x

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
    /**
     * Executes a Gremlin profile query on the Neptune Analytics graph.
     *
     * @param client       the {@link NeptuneGraphClient} instance to use for the
query
     * @param graphId      the identifier of the graph to execute the query on
     *
     * @throws NeptuneGraphException if an error occurs while executing the query on
the Neptune Graph
     * @throws Exception if an unexpected error occurs
```

```
    */
   public static void executeGremlinProfileQuery(NeptuneGraphClient client, String
graphId) {

       try {
           System.out.println("Running openCypher query on Neptune Analytics...");

           ExecuteQueryRequest request = ExecuteQueryRequest.builder()
                   .graphIdentifier(graphId)
                   .queryString("MATCH (n {code: 'ANC'}) RETURN n")
                   .language("OPEN_CYPHER")
                   .build();

           ResponseInputStream<ExecuteQueryResponse> response =
client.executeQuery(request);
           try (BufferedReader reader = new BufferedReader(new
InputStreamReader(response, StandardCharsets.UTF_8))) {
               String result = reader.lines().collect(Collectors.joining("\n"));
               System.out.println("Query Result:");
               System.out.println(result);
           } catch (Exception e) {
               System.err.println("Error reading response: " + e.getMessage());
           }

       } catch (NeptuneGraphException e) {
           System.err.println("NeptuneGraph error: " +
e.awsErrorDetails().errorMessage());
       } catch (Exception e) {
           System.err.println("Unexpected error: " + e.getMessage());
       } finally {
           client.close();
       }
   }
```

- Per i dettagli sull'API, consulta la ExecuteQuerysezione AWS SDK for Java 2.x API Reference.

`StartDBCluster`

Il seguente esempio di codice mostra come utilizzare`StartDBCluster`.

SDK per Java 2.x

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
   /**
    * Starts an Amazon Neptune DB cluster.
    *
    * @param clusterIdentifier the unique identifier of the DB cluster to be
stopped
    */
   public CompletableFuture<StartDbClusterResponse> startDBClusterAsync(String
clusterIdentifier) {
       StartDbClusterRequest clusterRequest = StartDbClusterRequest.builder()
               .dbClusterIdentifier(clusterIdentifier)
               .build();

       return getAsyncClient().startDBCluster(clusterRequest)
               .whenComplete((response, error) -> {
                   if (error != null) {
                       Throwable cause = error.getCause() != null ?
error.getCause() : error;

                       if (cause instanceof ResourceNotFoundException) {
                           throw (ResourceNotFoundException) cause;
                       }

                       throw new RuntimeException("Failed to start DB cluster: " +
cause.getMessage(), cause);
                   } else {
                       logger.info("DB Cluster starting: " + clusterIdentifier);
                   }
               });
   }
```

- Per i dettagli sull'API, consulta Start DBCluster in AWS SDK for Java 2.x API Reference.

`StopDBCluster`

Il seguente esempio di codice mostra come utilizzare`StopDBCluster`.

SDK per Java 2.x

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
   /**
    * Stops an Amazon Neptune DB cluster.
    *
    * @param clusterIdentifier the unique identifier of the DB cluster to be
stopped
    */
   public CompletableFuture<StopDbClusterResponse> stopDBClusterAsync(String
clusterIdentifier) {
       StopDbClusterRequest clusterRequest = StopDbClusterRequest.builder()
               .dbClusterIdentifier(clusterIdentifier)
               .build();

       return getAsyncClient().stopDBCluster(clusterRequest)
               .whenComplete((response, error) -> {
                   if (error != null) {
                       Throwable cause = error.getCause() != null ?
error.getCause() : error;

                       if (cause instanceof ResourceNotFoundException) {
                           throw (ResourceNotFoundException) cause;
                       }

                       throw new RuntimeException("Failed to stop DB cluster: " +
cause.getMessage(), cause);
                   } else {
                       logger.info("DB Cluster stopped: " + clusterIdentifier);
                   }
               });
   }
```

- Per i dettagli sull'API, consulta Stop DBCluster in AWS SDK for Java 2.x API Reference.

Scenari

Usa l'API Neptune per interrogare i dati del grafico

Il seguente esempio di codice mostra come utilizzare l'API Neptune per interrogare i dati del grafico.

SDK per Java 2.x

Mostra come usare l'API Java di Amazon Neptune per creare una funzione Lambda che interroga i dati del grafico all'interno del VPC.

Per il codice sorgente completo e le istruzioni su come configurarlo ed eseguirlo, consulta l'esempio completo su. GitHub

Servizi utilizzati in questo esempio

- Lambda
- Neptune

Esempi di Partner Central con SDK for Java 2.x

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando Partner Central. 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.

Gli scenari sono esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio o combinate con altri Servizi AWS.

Ogni esempio include un collegamento al codice sorgente completo, in cui è possibile trovare istruzioni su come configurare ed eseguire il codice nel contesto.

Argomenti

- Azioni

- Scenari

Azioni

`AssignOpportunity`

Il seguente esempio di codice mostra come utilizzare`AssignOpportunity`.

SDK per Java 2.x

Riassegna un'opportunità esistente a un altro utente.

```
package org.example;

import static org.example.utils.Constants.*;

import org.example.utils.Constants;
import org.example.utils.ReferenceCodesUtils;

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.partnercentralselling.PartnerCentralSellingClient;
import
 software.amazon.awssdk.services.partnercentralselling.model.AssignOpportunityRequest;
import
 software.amazon.awssdk.services.partnercentralselling.model.AssignOpportunityResponse;
import software.amazon.awssdk.services.partnercentralselling.model.AssigneeContact;

/*
Purpose
PC-API-07 Assigning a new owner
*/

public class AssignOpportunity {

 static PartnerCentralSellingClient client = PartnerCentralSellingClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(DefaultCredentialsProvider.create())
            .httpClient(ApacheHttpClient.builder().build())
            .build();
```

```
    public static void main(String[] args) {

     String opportunityId = args.length > 0 ? args[0] : OPPORTUNITY_ID;

     String assigneeFirstName = "John";

     String assigneeLastName = "Doe";

     String assigneeEmail = "test@test.com";

     String businessTitle = "PartnerAccountManager";

     AssignOpportunityResponse response = getResponse(opportunityId,
 assigneeFirstName, assigneeLastName, assigneeEmail, businessTitle);

     ReferenceCodesUtils.formatOutput(response);
    }

 static AssignOpportunityResponse getResponse(String opportunityId, String
 assigneeFirstName, String assigneeLastName, String assigneeEmail, String
 businessTitle) {

  AssignOpportunityRequest assignOpportunityRequest =
 AssignOpportunityRequest.builder()
    .catalog(Constants.CATALOG_TO_USE)
          .identifier(opportunityId)
          .assignee(AssigneeContact.builder()
            .firstName(assigneeFirstName)
            .lastName(assigneeLastName)
            .email(assigneeEmail)
            .businessTitle(businessTitle)
            .build())
          .build();

        AssignOpportunityResponse response =
 client.assignOpportunity(assignOpportunityRequest);

        return response;
 }
}
```

- Per i dettagli sull'API, consulta la sezione AssignOpportunity AWS SDK for Java 2.xAPI Reference.

`AssociateOpportunity`

Il seguente esempio di codice mostra come utilizzare`AssociateOpportunity`.

SDK per Java 2.x

Crea un'associazione formale tra un'opportunità e varie entità correlate.

```
package org.example;

import static org.example.utils.Constants.*;

import org.example.utils.Constants;
import org.example.utils.ReferenceCodesUtils;

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.partnercentralselling.PartnerCentralSellingClient;
import
 software.amazon.awssdk.services.partnercentralselling.model.AssociateOpportunityRequest;
import
 software.amazon.awssdk.services.partnercentralselling.model.AssociateOpportunityResponse;

/*
Purpose
PC-API -11 Associating a product
PC-API -12 Associating a solution
PC-API -13 Associating an offer
entity_type = Solutions | AWSProducts | AWSMarketplaceOffers
*/

public class AssociateOpportunity {

 static PartnerCentralSellingClient client = PartnerCentralSellingClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(DefaultCredentialsProvider.create())
            .httpClient(ApacheHttpClient.builder().build())
            .build();
```

```
    public static void main(String[] args) {

     String opportunityId = args.length > 0 ? args[0] : OPPORTUNITY_ID;

     String entityType = "Solutions";

     String entityIdentifier = "S-0000000";

     AssociateOpportunityResponse response = getResponse(opportunityId, entityType,
 entityIdentifier );

     ReferenceCodesUtils.formatOutput(response);
    }

static AssociateOpportunityResponse getResponse(String opportunityId, String
 entityType, String entityIdentifier) {

        AssociateOpportunityRequest associateOpportunityRequest =
 AssociateOpportunityRequest.builder()
    .catalog(Constants.CATALOG_TO_USE)
          .opportunityIdentifier(opportunityId)
          .relatedEntityType(entityType)
          .relatedEntityIdentifier(entityIdentifier)
          .build();

        AssociateOpportunityResponse response =
 client.associateOpportunity(associateOpportunityRequest);

        return response;
 }
}
```

- Per i dettagli sull'API, consulta la AssociateOpportunitysezione AWS SDK for Java 2.x API Reference.

`CreateOpportunity`

Il seguente esempio di codice mostra come utilizzare`CreateOpportunity`.

SDK per Java 2.x

Crea un'opportunità.

```
package org.example;

import java.time.Instant;
import java.util.ArrayList;
import java.util.List;

import static org.example.utils.Constants.*;

import org.example.entity.Root;
import org.example.utils.ReferenceCodesUtils;
import org.example.utils.StringSerializer;

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.partnercentralselling.PartnerCentralSellingClient;
import software.amazon.awssdk.services.partnercentralselling.model.Account;
import software.amazon.awssdk.services.partnercentralselling.model.Address;
import software.amazon.awssdk.services.partnercentralselling.model.Contact;
import
 software.amazon.awssdk.services.partnercentralselling.model.CreateOpportunityRequest;
import
 software.amazon.awssdk.services.partnercentralselling.model.CreateOpportunityResponse;
import software.amazon.awssdk.services.partnercentralselling.model.Customer;
import
 software.amazon.awssdk.services.partnercentralselling.model.ExpectedCustomerSpend;
import software.amazon.awssdk.services.partnercentralselling.model.LifeCycle;
import software.amazon.awssdk.services.partnercentralselling.model.Marketing;
import software.amazon.awssdk.services.partnercentralselling.model.MonetaryValue;
import software.amazon.awssdk.services.partnercentralselling.model.NextStepsHistory;
import software.amazon.awssdk.services.partnercentralselling.model.Project;
import software.amazon.awssdk.services.partnercentralselling.model.SoftwareRevenue;

import com.google.gson.Gson;
import com.google.gson.GsonBuilder;
import com.google.gson.ToNumberPolicy;

public class CreateOpportunity {
```

```
static final Gson GSON = new GsonBuilder()
  .setObjectToNumberStrategy(ToNumberPolicy.LAZILY_PARSED_NUMBER)
  .registerTypeAdapter(String.class, new StringSerializer())
  .create();

static PartnerCentralSellingClient client = PartnerCentralSellingClient.builder()
           .region(Region.US_EAST_1)
           .credentialsProvider(DefaultCredentialsProvider.create())
           .httpClient(ApacheHttpClient.builder().build())
           .build();

public static void main(String[] args) {

 String inputFile = "CreateOpportunity2.json";

 if (args.length > 0)
  inputFile = args[0];

 CreateOpportunityResponse response = createOpportunity(inputFile);

 client.close();
}

static CreateOpportunityResponse createOpportunity(String inputFile) {

 String inputString = ReferenceCodesUtils.readInputFileToString(inputFile);

 Root root = GSON.fromJson(inputString, Root.class);

 List<NextStepsHistory> nextStepsHistories = new ArrayList<NextStepsHistory>();
 if ( root.lifeCycle != null && root.lifeCycle.nextStepsHistories != null) {
  for (org.example.entity.NextStepsHistory nextStepsHistoryJson :
root.lifeCycle.nextStepsHistories) {
   NextStepsHistory nextStepsHistory = NextStepsHistory.builder()
     .time(Instant.parse(nextStepsHistoryJson.time))
     .value(nextStepsHistoryJson.value)
                 .build();
   nextStepsHistories.add(nextStepsHistory);
  }
 }

 LifeCycle lifeCycle = null;
 if ( root.lifeCycle != null ) {
  lifeCycle = LifeCycle.builder()
```

```
    .closedLostReason(root.lifeCycle.closedLostReason)
    .nextSteps(root.lifeCycle.nextSteps)
    .nextStepsHistory(nextStepsHistories)
    .reviewComments(root.lifeCycle.reviewComments)
    .reviewStatus(root.lifeCycle.reviewStatus)
    .reviewStatusReason(root.lifeCycle.reviewStatusReason)
    .stage(root.lifeCycle.stage)
    .targetCloseDate(root.lifeCycle.targetCloseDate)
    .build();
 }

 Marketing marketing = null;
 if ( root.marketing != null ) {
  marketing = Marketing.builder()
     .awsFundingUsed(root.marketing.awsFundingUsed)
     .campaignName(root.marketing.campaignName)
     .channels(root.marketing.channels)
     .source(root.marketing.source)
     .useCases(root.marketing.useCases)
     .build();

 }

 Address address = null;
 if ( root.customer != null && root.customer.account != null &&
root.customer.account.address != null ) {
  address = Address.builder()
   .city(root.customer.account.address.city)
               .postalCode(root.customer.account.address.postalCode)
               .stateOrRegion(root.customer.account.address.stateOrRegion)
               .countryCode(root.customer.account.address.countryCode)
               .streetAddress(root.customer.account.address.streetAddress)
               .build();
 }

 Account account = null;
 if ( root.customer != null && root.customer.account!= null) {
  account = Account.builder()
            .address(address)
            .awsAccountId(root.customer.account.awsAccountId)
               .duns(root.customer.account.duns)
               .industry(root.customer.account.industry)
               .otherIndustry(root.customer.account.otherIndustry)
               .companyName(root.customer.account.companyName)
```

```
                .websiteUrl(root.customer.account.websiteUrl)
                .build();
 }

 List<Contact> contacts = new ArrayList<Contact>();
 if ( root.customer != null && root.customer.contacts != null) {
  for (org.example.entity.Contact jsonContact : root.customer.contacts) {
   Contact contact = Contact.builder()
                .email(jsonContact.email)
                .firstName(jsonContact.firstName)
                .lastName(jsonContact.lastName)
                .phone(jsonContact.phone)
                .businessTitle(jsonContact.businessTitle)
                .build();
   contacts.add(contact);
  }
 }

 Customer customer = Customer.builder()
   .account(account)
   .contacts(contacts)
   .build();

 Contact oportunityTeamContact = null;
 if (root.opportunityTeam != null && root.opportunityTeam.get(0) != null ) {
  oportunityTeamContact = Contact.builder()
                .firstName(root.opportunityTeam.get(0).firstName)
                .lastName(root.opportunityTeam.get(0).lastName)
                .email(root.opportunityTeam.get(0).email)
                .phone(root.opportunityTeam.get(0).phone)
                .businessTitle(root.opportunityTeam.get(0).businessTitle)
                .build();
 }

 List<ExpectedCustomerSpend> expectedCustomerSpends = new
ArrayList<ExpectedCustomerSpend>();
 if ( root.project != null && root.project.expectedCustomerSpend != null) {
  for (org.example.entity.ExpectedCustomerSpend expectedCustomerSpendJson :
root.project.expectedCustomerSpend) {
   ExpectedCustomerSpend expectedCustomerSpend = null;
   expectedCustomerSpend = ExpectedCustomerSpend.builder()
     .amount(expectedCustomerSpendJson.amount)
     .currencyCode(expectedCustomerSpendJson.currencyCode)
     .frequency(expectedCustomerSpendJson.frequency)
```

```
      .targetCompany(expectedCustomerSpendJson.targetCompany)
      .build();
    expectedCustomerSpends.add(expectedCustomerSpend);
   }
        }

        Project project = null;
        if ( root.project != null) {
         project = Project.builder()
                .title(root.project.title)
                .customerBusinessProblem(root.project.customerBusinessProblem)
                .customerUseCase(root.project.customerUseCase)
                .deliveryModels(root.project.deliveryModels)
                .expectedCustomerSpend(expectedCustomerSpends)
                .salesActivities(root.project.salesActivities)
                .competitorName(root.project.competitorName)
                .otherSolutionDescription(root.project.otherSolutionDescription)
                .build();
        }

        SoftwareRevenue softwareRevenue = null;
        if ( root.softwareRevenue != null) {
         MonetaryValue monetaryValue = null;
         if ( root.softwareRevenue.value != null) {
          monetaryValue = MonetaryValue.builder()
             .amount(root.softwareRevenue.value.amount)
             .currencyCode(root.softwareRevenue.value.currencyCode)
             .build();
         }
         softwareRevenue = SoftwareRevenue.builder()
           .deliveryModel(root.softwareRevenue.deliveryModel)
           .effectiveDate(root.softwareRevenue.effectiveDate)
           .expirationDate(root.softwareRevenue.expirationDate)
           .value(monetaryValue)
           .build();
        }

 // Building the Actual CreateOpportunity Request
 CreateOpportunityRequest createOpportunityRequest =
CreateOpportunityRequest.builder()
   .catalog(CATALOG_TO_USE)
   .clientToken(root.clientToken)
   .primaryNeedsFromAwsWithStrings(root.primaryNeedsFromAws)
   .opportunityType(root.opportunityType)
```

```
    .lifeCycle(lifeCycle)
    .marketing(marketing)
    .nationalSecurity(root.nationalSecurity)
    .origin(root.origin)
    .customer(customer)
    .project(project)
    .partnerOpportunityIdentifier(root.partnerOpportunityIdentifier)
    .opportunityTeam(oportunityTeamContact)
    .softwareRevenue(softwareRevenue)
    .build();

  CreateOpportunityResponse response =
 client.createOpportunity(createOpportunityRequest);
  System.out.println("Successfully created: " + response);

  return response;
    }

}
```

- Per i dettagli sull'API, consulta la CreateOpportunitysezione AWS SDK for Java 2.x API Reference.

`DisassociateOpportunity`

Il seguente esempio di codice mostra come utilizzare`DisassociateOpportunity`.

SDK per Java 2.x

Rimuove un'associazione esistente tra un'opportunità e le entità correlate.

```
package org.example;

import static org.example.utils.Constants.*;

import org.example.utils.Constants;
import org.example.utils.ReferenceCodesUtils;

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.regions.Region;
```

```
import
 software.amazon.awssdk.services.partnercentralselling.PartnerCentralSellingClient;
import
 software.amazon.awssdk.services.partnercentralselling.model.DisassociateOpportunityRequest;
import
 software.amazon.awssdk.services.partnercentralselling.model.DisassociateOpportunityResponse

/*
Purpose
PC-API -14 Removing a Solution
PC-API -15 Removing an offer
PC-API -16 Removing a product
entity_type = Solutions | AWSProducts | AWSMarketplaceOffers
*/

public class DisassociateOpportunity {

 static PartnerCentralSellingClient client = PartnerCentralSellingClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(DefaultCredentialsProvider.create())
            .httpClient(ApacheHttpClient.builder().build())
            .build();

    public static void main(String[] args) {

     String opportunityId = args.length > 0 ? args[0] : OPPORTUNITY_ID;

     String entityType = "Solutions";

     String entityIdentifier = "S-0000000";

     DisassociateOpportunityResponse response = getResponse(opportunityId,
 entityType, entityIdentifier );

     ReferenceCodesUtils.formatOutput(response);
    }

 static DisassociateOpportunityResponse getResponse(String opportunityId, String
 entityType, String entityIdentifier) {

  DisassociateOpportunityRequest disassociateOpportunityRequest =
 DisassociateOpportunityRequest.builder()
    .catalog(Constants.CATALOG_TO_USE)
          .opportunityIdentifier(opportunityId)
```

```
            .relatedEntityType(entityType)
            .relatedEntityIdentifier(entityIdentifier)
            .build();

        DisassociateOpportunityResponse response =
 client.disassociateOpportunity(disassociateOpportunityRequest);

        return response;
 }
}
```

- Per i dettagli sull'API, consulta la DisassociateOpportunitysezione AWS SDK for Java 2.x API Reference.

`GetAwsOpportunitySummary`

Il seguente esempio di codice mostra come utilizzare`GetAwsOpportunitySummary`.

SDK per Java 2.x

Recupera un riepilogo di un' AWS opportunità.

```
package org.example;

import static org.example.utils.Constants.*;

import org.example.utils.Constants;
import org.example.utils.ReferenceCodesUtils;

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.partnercentralselling.PartnerCentralSellingClient;
import
 software.amazon.awssdk.services.partnercentralselling.model.GetAwsOpportunitySummaryRequest
import
 software.amazon.awssdk.services.partnercentralselling.model.GetAwsOpportunitySummaryRespons

/*
 * Purpose
 * PC-API-25 Retrieves a summary of an AWS Opportunity.
```

```
 */

public class GetAwsOpportunitySummary {

 static PartnerCentralSellingClient client = PartnerCentralSellingClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(DefaultCredentialsProvider.create())
            .httpClient(ApacheHttpClient.builder().build())
            .build();

    public static void main(String[] args) {

     String opportunityId = args.length > 0 ? args[0] : OPPORTUNITY_ID;

     GetAwsOpportunitySummaryResponse response = getResponse(opportunityId);

     ReferenceCodesUtils.formatOutput(response);
    }

 public static GetAwsOpportunitySummaryResponse getResponse(String opportunityId) {

  GetAwsOpportunitySummaryRequest getOpportunityRequest =
 GetAwsOpportunitySummaryRequest.builder()
    .catalog(Constants.CATALOG_TO_USE)
          .relatedOpportunityIdentifier(opportunityId)
          .build();

  GetAwsOpportunitySummaryResponse response =
 client.getAwsOpportunitySummary(getOpportunityRequest);

        return response;
 }
}
```

- Per i dettagli sull'API, consulta la sezione GetAwsOpportunitySummary AWS SDK for Java 2.xAPI Reference.

`GetEngagementInvitation`

Il seguente esempio di codice mostra come utilizzare`GetEngagementInvitation`.

SDK per Java 2.x

Recupera i dettagli di un invito di fidanzamento condiviso AWS da un partner.

```
package org.example;

import static org.example.utils.Constants.*;

import org.example.utils.Constants;
import org.example.utils.ReferenceCodesUtils;

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.partnercentralselling.PartnerCentralSellingClient;
import
 software.amazon.awssdk.services.partnercentralselling.model.GetEngagementInvitationRequest;
import
 software.amazon.awssdk.services.partnercentralselling.model.GetEngagementInvitationResponse

/*
 * Purpose
 * PC-API-22 Get engagement invitation opportunity
 */

public class GetEngagementInvitation {

 static PartnerCentralSellingClient client = PartnerCentralSellingClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(DefaultCredentialsProvider.create())
            .httpClient(ApacheHttpClient.builder().build())
            .build();

    public static void main(String[] args) {

     String opportunityId = args.length > 0 ? args[0] : OPPORTUNITY_ID;

     GetEngagementInvitationResponse response = getResponse(opportunityId);

     ReferenceCodesUtils.formatOutput(response);
    }

 static GetEngagementInvitationResponse getResponse(String opportunityId) {
```

```
 GetEngagementInvitationRequest getOpportunityRequest =
GetEngagementInvitationRequest.builder()
   .catalog(Constants.CATALOG_TO_USE)
         .identifier(opportunityId)
         .build();

 GetEngagementInvitationResponse response =
client.getEngagementInvitation(getOpportunityRequest);

       return response;
}
}
```

- Per i dettagli sull'API, consulta la sezione GetEngagementInvitation AWS SDK for Java 2.xAPI Reference.

`GetOpportunity`

Il seguente esempio di codice mostra come utilizzare`GetOpportunity`.

SDK per Java 2.x

Ottieni un'opportunità.

```
package org.example;

import static org.example.utils.Constants.*;

import org.example.utils.Constants;
import org.example.utils.ReferenceCodesUtils;

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.partnercentralselling.PartnerCentralSellingClient;
import
 software.amazon.awssdk.services.partnercentralselling.model.GetOpportunityRequest;
import
 software.amazon.awssdk.services.partnercentralselling.model.GetOpportunityResponse;
```

```
/*
 * Purpose
 * PC-API-08 Get updated Opportunity
 */

public class GetOpportunity {

 static PartnerCentralSellingClient client = PartnerCentralSellingClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(DefaultCredentialsProvider.create())
            .httpClient(ApacheHttpClient.builder().build())
            .build();

    public static void main(String[] args) {

     String opportunityId = args.length > 0 ? args[0] : OPPORTUNITY_ID;

     GetOpportunityResponse response = getResponse(opportunityId);

     ReferenceCodesUtils.formatOutput(response);
    }

 public static GetOpportunityResponse getResponse(String opportunityId) {

        GetOpportunityRequest getOpportunityRequest =
 GetOpportunityRequest.builder()
    .catalog(Constants.CATALOG_TO_USE)
          .identifier(opportunityId)
          .build();

        GetOpportunityResponse response =
 client.getOpportunity(getOpportunityRequest);

        return response;
 }
}
```

- Per i dettagli sulle API, consulta la GetOpportunitysezione AWS SDK for Java 2.x API Reference.

`ListEngagementInvitations`

Il seguente esempio di codice mostra come utilizzare`ListEngagementInvitations`.

SDK per Java 2.x

Recupera un elenco di inviti di fidanzamento inviati al partner.

```
package org.example;

import java.util.ArrayList;
import java.util.List;

import org.example.utils.ReferenceCodesUtils;
import static org.example.utils.Constants.*;

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.partnercentralselling.PartnerCentralSellingClient;
import
 software.amazon.awssdk.services.partnercentralselling.model.ListEngagementInvitationsReques
import
 software.amazon.awssdk.services.partnercentralselling.model.ListEngagementInvitationsRespon
import software.amazon.awssdk.services.partnercentralselling.model.ParticipantType;
import
 software.amazon.awssdk.services.partnercentralselling.model.EngagementInvitationSummary;

public class ListEngagementInvitations {

 static PartnerCentralSellingClient client = PartnerCentralSellingClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(DefaultCredentialsProvider.create())
            .httpClient(ApacheHttpClient.builder().build())
            .build();

    public static void main(String[] args) {

     List<EngagementInvitationSummary> opportunitySummaries = getResponse();
        ReferenceCodesUtils.formatOutput(opportunitySummaries);
    }

    static List<EngagementInvitationSummary> getResponse() {
```

```
 List<EngagementInvitationSummary> opportunitySummaries = new
ArrayList<EngagementInvitationSummary>();

 ListEngagementInvitationsRequest listOpportunityRequest =
ListEngagementInvitationsRequest.builder()
               .catalog(CATALOG_TO_USE)
               .participantType(ParticipantType.RECEIVER)
         .maxResults(5)
         .build();

 ListEngagementInvitationsResponse response =
client.listEngagementInvitations(listOpportunityRequest);

    opportunitySummaries.addAll(response.engagementInvitationSummaries());

    client.close();

       return opportunitySummaries;
 }
}
```

- Per i dettagli sull'API, consulta la sezione AWS SDK for Java 2.x API ListEngagementInvitationsReference.

`ListOpportunities`

Il seguente esempio di codice mostra come utilizzare`ListOpportunities`.

SDK per Java 2.x

Elenca le opportunità.

```
package org.example;

import java.util.ArrayList;
import java.util.List;

import org.example.utils.ReferenceCodesUtils;
import static org.example.utils.Constants.*;

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
```

```
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.partnercentralselling.PartnerCentralSellingClient;
import
 software.amazon.awssdk.services.partnercentralselling.model.ListOpportunitiesRequest;
import
 software.amazon.awssdk.services.partnercentralselling.model.ListOpportunitiesResponse;
import
 software.amazon.awssdk.services.partnercentralselling.model.OpportunitySummary;

/*
 * Purpose
 * PC-API-18 Getting list of Opportunities
 */

public class ListOpportunititesPaging {

 static PartnerCentralSellingClient client = PartnerCentralSellingClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(DefaultCredentialsProvider.create())
            .httpClient(ApacheHttpClient.builder().build())
            .build();

    public static void main(String[] args) {
     List<OpportunitySummary> opportunitySummaries = getResponse();
        ReferenceCodesUtils.formatOutput(opportunitySummaries);
    }

    private static List<OpportunitySummary> getResponse() {
     List<OpportunitySummary> opportunitySummaries = new
 ArrayList<OpportunitySummary>();

  ListOpportunitiesRequest listOpportunityRequest =
 ListOpportunitiesRequest.builder()
                .catalog(CATALOG_TO_USE)
          .maxResults(5)
          .build();

     ListOpportunitiesResponse response =
 client.listOpportunities(listOpportunityRequest);

     opportunitySummaries.addAll(response.opportunitySummaries());
```

```
     while (response.nextToken() != null && response.nextToken().length() > 0) {
      listOpportunityRequest = ListOpportunitiesRequest.builder()
                    .catalog(CATALOG_TO_USE)
              .maxResults(5)
              .nextToken(response.nextToken())
              .build();
      response = client.listOpportunities(listOpportunityRequest);
      opportunitySummaries.addAll(response.opportunitySummaries());
     }

     client.close();

        return opportunitySummaries;
 }
}
```

- Per i dettagli sull'API, ListOpportunitiesconsulta AWS SDK for Java 2.x API Reference.

`ListSolutions`

Il seguente esempio di codice mostra come utilizzare`ListSolutions`.

SDK per Java 2.x

Recupera un elenco di soluzioni Partner che il partner ha registrato su Partner Central.

```
package org.example;

import java.util.ArrayList;
import java.util.List;

import static org.example.utils.Constants.*;
import org.example.utils.ReferenceCodesUtils;

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.partnercentralselling.PartnerCentralSellingClient;
import
 software.amazon.awssdk.services.partnercentralselling.model.ListSolutionsRequest;
```

```
import
 software.amazon.awssdk.services.partnercentralselling.model.ListSolutionsResponse;
import software.amazon.awssdk.services.partnercentralselling.model.SolutionBase;

/*
 * Purpose
 * PC-API-10 Getting list of solutions
 */

public class ListSolutions {

 static PartnerCentralSellingClient client = PartnerCentralSellingClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(DefaultCredentialsProvider.create())
            .httpClient(ApacheHttpClient.builder().build())
            .build();

    public static void main(String[] args) {
     List<SolutionBase> solutionSummaries = getResponse();
        ReferenceCodesUtils.formatOutput(solutionSummaries);
    }

    static List<SolutionBase> getResponse() {
  List<SolutionBase> solutionSummaries = new ArrayList<SolutionBase>();

  ListSolutionsRequest listSolutionsRequest = ListSolutionsRequest.builder()
    .catalog(CATALOG_TO_USE)
          .maxResults(5)
          .build();

     ListSolutionsResponse response = client.listSolutions(listSolutionsRequest);

     solutionSummaries.addAll(response.solutionSummaries());

        return solutionSummaries;
 }
}
```

- Per i dettagli sulle API, consulta la sezione ListSolutions AWS SDK for Java 2.xAPI Reference.

`RejectEngagementInvitation`

Il seguente esempio di codice mostra come utilizzare`RejectEngagementInvitation`.

SDK per Java 2.x

Rifiuta qualsiasi cosa EngagementInvitation AWS condivisa.

```
package org.example;

import static org.example.utils.Constants.*;

import org.example.utils.Constants;
import org.example.utils.ReferenceCodesUtils;

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.partnercentralselling.PartnerCentralSellingClient;
import
 software.amazon.awssdk.services.partnercentralselling.model.RejectEngagementInvitationReque
import
 software.amazon.awssdk.services.partnercentralselling.model.RejectEngagementInvitationRespo

/*
 * Purpose
 * PC-API-05 AWS Originated(AO) rejection
 */

public class RejectEngagementInvitation {

 static PartnerCentralSellingClient client = PartnerCentralSellingClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(DefaultCredentialsProvider.create())
            .httpClient(ApacheHttpClient.builder().build())
            .build();

    public static void main(String[] args) {

     String opportunityId = args.length > 0 ? args[0] : OPPORTUNITY_ID;

  RejectEngagementInvitationResponse response = getResponse(opportunityId);
```

```
     ReferenceCodesUtils.formatOutput(response);
    }

 static RejectEngagementInvitationResponse getResponse(String invitationId) {

        RejectEngagementInvitationRequest rejectOpportunityRequest =
 RejectEngagementInvitationRequest.builder()
    .catalog(Constants.CATALOG_TO_USE)
          .identifier(invitationId)
          .rejectionReason("Unable to support")
          .build();

  RejectEngagementInvitationResponse response =
 client.rejectEngagementInvitation(rejectOpportunityRequest);

        return response;
 }
}
```

- Per i dettagli sull'API, consulta la sezione RejectEngagementInvitation AWS SDK for Java 2.xAPI Reference.

`StartEngagementByAcceptingInvitationTask`

Il seguente esempio di codice mostra come utilizzare`StartEngagementByAcceptingInvitationTask`.

SDK per Java 2.x

Inizia il fidanzamento accettando un EngagementInvitation.

```
package org.example;

import static org.example.utils.Constants.*;

import org.example.utils.Constants;
import org.example.utils.ReferenceCodesUtils;

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.regions.Region;
```

```
import
 software.amazon.awssdk.services.partnercentralselling.PartnerCentralSellingClient;
import
 software.amazon.awssdk.services.partnercentralselling.model.StartEngagementByAcceptingInvit
import
 software.amazon.awssdk.services.partnercentralselling.model.StartEngagementByAcceptingInvit
import
 software.amazon.awssdk.services.partnercentralselling.model.GetEngagementInvitationRequest;
import
 software.amazon.awssdk.services.partnercentralselling.model.GetEngagementInvitationResponse
import software.amazon.awssdk.services.partnercentralselling.model.InvitationStatus;

/*
Purpose
PC-API-04: Start Engagement By Accepting InvitationTask for AWS Originated(AO)
 opportunity
*/

public class StartEngagementByAcceptingInvitationTask {

 static PartnerCentralSellingClient client = PartnerCentralSellingClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(DefaultCredentialsProvider.create())
            .httpClient(ApacheHttpClient.builder().build())
            .build();

 static String clientToken = "test-a30d161";

    public static void main(String[] args) {

     String opportunityId = args.length > 0 ? args[0] : OPPORTUNITY_ID;

     StartEngagementByAcceptingInvitationTaskResponse response =
 getResponse(opportunityId);

     if ( response == null) {
      System.out.println("Opportunity is not AWS Originated.");
     } else {
      ReferenceCodesUtils.formatOutput(response);
     }
    }

    private static GetEngagementInvitationResponse getInvitation(String
 invitationId) {
```

```
    GetEngagementInvitationRequest getRequest =
GetEngagementInvitationRequest.builder()
         .catalog(Constants.CATALOG_TO_USE)
         .identifier(invitationId)
         .build();

 GetEngagementInvitationResponse response =
client.getEngagementInvitation(getRequest);

       return response;
}

static StartEngagementByAcceptingInvitationTaskResponse getResponse(String
invitationId) {

 if ( getInvitation(invitationId).status().equals(InvitationStatus.PENDING)) {
  StartEngagementByAcceptingInvitationTaskRequest acceptOpportunityRequest =
    StartEngagementByAcceptingInvitationTaskRequest.builder()
    .catalog(Constants.CATALOG_TO_USE)
          .identifier(invitationId)
          .clientToken(clientToken)
          .build();

  StartEngagementByAcceptingInvitationTaskResponse response =
client.startEngagementByAcceptingInvitationTask(acceptOpportunityRequest);
        return response;
 }
 return null;
}
}
```

- Per i dettagli sulle API, consulta la StartEngagementByAcceptingInvitationTasksezione AWS SDK for Java 2.x API Reference.

`StartEngagementFromOpportunityTask`

Il seguente esempio di codice mostra come utilizzare`StartEngagementFromOpportunityTask`.

SDK per Java 2.x

Avvia il processo di coinvolgimento partendo da un'opportunità esistente accettando l'invito al coinvolgimento e creando un'opportunità corrispondente nel sistema del partner.

```
package org.example;

import static org.example.utils.Constants.*;

import org.example.utils.Constants;
import org.example.utils.ReferenceCodesUtils;

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.partnercentralselling.PartnerCentralSellingClient;
import software.amazon.awssdk.services.partnercentralselling.model.AwsSubmission;
import
 software.amazon.awssdk.services.partnercentralselling.model.SalesInvolvementType;
import
 software.amazon.awssdk.services.partnercentralselling.model.StartEngagementFromOpportunityT
import
 software.amazon.awssdk.services.partnercentralselling.model.StartEngagementFromOpportunityT
import software.amazon.awssdk.services.partnercentralselling.model.Visibility;

/*
 * Purpose
 * PC-API-01 Partner Originated (PO) opp submission(Start Engagement From
 Opportunity Task for AO Originated Opportunity)
 */

public class StartEngagementFromOpportunityTask {

 static PartnerCentralSellingClient client = PartnerCentralSellingClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(DefaultCredentialsProvider.create())
            .httpClient(ApacheHttpClient.builder().build())
            .build();

    public static void main(String[] args) {

     String opportunityId = args.length > 0 ? args[0] : OPPORTUNITY_ID;
```

```
    StartEngagementFromOpportunityTaskResponse response =
 getResponse(opportunityId);

    ReferenceCodesUtils.formatOutput(response);
   }

 static StartEngagementFromOpportunityTaskResponse getResponse(String opportunityId)
 {

  StartEngagementFromOpportunityTaskRequest submitOpportunityRequest =
 StartEngagementFromOpportunityTaskRequest.builder()
    .catalog(Constants.CATALOG_TO_USE)
          .identifier(opportunityId)
          .clientToken("test-annjqwesdsd99")

 .awsSubmission(AwsSubmission.builder().involvementType(SalesInvolvementType.CO_SELL).visibi
          .build();

  StartEngagementFromOpportunityTaskResponse response =
 client.startEngagementFromOpportunityTask(submitOpportunityRequest);

        return response;
 }
}
```

- Per i dettagli sulle API, consulta la sezione StartEngagementFromOpportunityTask AWS SDK for Java 2.xAPI Reference.

`UpdateOpportunity`

Il seguente esempio di codice mostra come utilizzare`UpdateOpportunity`.

SDK per Java 2.x

Aggiornare un'opportunità.

```
package org.example;

import java.time.Instant;
import java.util.ArrayList;
import java.util.List;
```

```
import static org.example.utils.Constants.*;

import org.example.entity.Root;
import org.example.utils.Constants;
import org.example.utils.ReferenceCodesUtils;
import org.example.utils.StringSerializer;

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.partnercentralselling.PartnerCentralSellingClient;
import software.amazon.awssdk.services.partnercentralselling.model.Account;
import software.amazon.awssdk.services.partnercentralselling.model.Address;
import software.amazon.awssdk.services.partnercentralselling.model.Contact;
import software.amazon.awssdk.services.partnercentralselling.model.Customer;
import
 software.amazon.awssdk.services.partnercentralselling.model.ExpectedCustomerSpend;
import
 software.amazon.awssdk.services.partnercentralselling.model.GetOpportunityRequest;
import
 software.amazon.awssdk.services.partnercentralselling.model.GetOpportunityResponse;
import software.amazon.awssdk.services.partnercentralselling.model.LifeCycle;
import software.amazon.awssdk.services.partnercentralselling.model.Marketing;
import software.amazon.awssdk.services.partnercentralselling.model.NextStepsHistory;
import software.amazon.awssdk.services.partnercentralselling.model.Project;
import software.amazon.awssdk.services.partnercentralselling.model.ReviewStatus;
import
 software.amazon.awssdk.services.partnercentralselling.model.UpdateOpportunityRequest;
import
 software.amazon.awssdk.services.partnercentralselling.model.UpdateOpportunityResponse;

import com.google.gson.Gson;
import com.google.gson.GsonBuilder;
import com.google.gson.ToNumberPolicy;

/*
 * Purpose
 * PC-API-02/06 Update opportunity when LifeCycle.ReviewStatus is not Submitted or
 In-Review
 */

public class UpdateOpportunity {
```

```
static final Gson GSON = new GsonBuilder()
  .setObjectToNumberStrategy(ToNumberPolicy.LAZILY_PARSED_NUMBER)
  .registerTypeAdapter(String.class, new StringSerializer())
  .create();

static PartnerCentralSellingClient client = PartnerCentralSellingClient.builder()
           .region(Region.US_EAST_1)
           .credentialsProvider(DefaultCredentialsProvider.create())
           .httpClient(ApacheHttpClient.builder().build())
           .build();

static String OPPORTUNITY_ORIGIN = ORIGIN_PARTNER_ORIGINATED;

public static void main(String[] args) {

 String inputFile = "updateOpportunity.json";

 if (args.length > 0)
  inputFile = args[0];

 UpdateOpportunityResponse response = updateOpportunity(inputFile);

 client.close();
}

public static GetOpportunityResponse getResponse(String opportunityId) {

       GetOpportunityRequest getOpportunityRequest =
GetOpportunityRequest.builder()
   .catalog(Constants.CATALOG_TO_USE)
         .identifier(opportunityId)
         .build();

       GetOpportunityResponse response =
client.getOpportunity(getOpportunityRequest);
       System.out.println(opportunityId + ":" + response);
       return response;
}

public static UpdateOpportunityResponse updateOpportunity(String inputFile) {

 String inputString = ReferenceCodesUtils.readInputFileToString(inputFile);

 Root root = GSON.fromJson(inputString, Root.class);
```

```
 GetOpportunityResponse response = getResponse(root.identifier);

 if (response != null
   && response.lifeCycle() != null
   && response.lifeCycle().reviewStatus() != null
   && response.lifeCycle().reviewStatus() != ReviewStatus.SUBMITTED
   && response.lifeCycle().reviewStatus() != ReviewStatus.IN_REVIEW) {

  List<NextStepsHistory> nextStepsHistories = new ArrayList<NextStepsHistory>();
  if ( root.lifeCycle != null && root.lifeCycle.nextStepsHistories != null) {
   for (org.example.entity.NextStepsHistory nextStepsHistoryJson :
root.lifeCycle.nextStepsHistories) {
    NextStepsHistory nextStepsHistory = NextStepsHistory.builder()
      .time(Instant.parse(nextStepsHistoryJson.time))
      .value(nextStepsHistoryJson.value)
                  .build();
    nextStepsHistories.add(nextStepsHistory);
   }
  }

  LifeCycle lifeCycle = null;
  if ( root.lifeCycle != null ) {
   lifeCycle = LifeCycle.builder()
    .closedLostReason(root.lifeCycle.closedLostReason)
    .nextSteps(root.lifeCycle.nextSteps)
    .nextStepsHistory(nextStepsHistories)
    .reviewComments(root.lifeCycle.reviewComments)
    .reviewStatus(root.lifeCycle.reviewStatus)
    .reviewStatusReason(root.lifeCycle.reviewStatusReason)
    .stage(root.lifeCycle.stage)
    .targetCloseDate(root.lifeCycle.targetCloseDate)
    .build();
  }

  Marketing marketing = null;
  if ( root.marketing != null ) {
   marketing = Marketing.builder()
     .awsFundingUsed(root.marketing.awsFundingUsed)
     .campaignName(root.marketing.campaignName)
     .channels(root.marketing.channels)
     .source(root.marketing.source)
     .useCases(root.marketing.useCases)
     .build();
```

```
  }

  Address address = null;
  if (root.customer != null && root.customer.account != null &&
root.customer.account.address != null) {
   address = Address.builder().postalCode(root.customer.account.address.postalCode)
     .stateOrRegion(root.customer.account.address.stateOrRegion)
     .countryCode(root.customer.account.address.countryCode).build();
  }

  Account account = null;
  if (root.customer != null && root.customer.account != null) {
   account = Account.builder().address(address).duns(root.customer.account.duns)

.industry(root.customer.account.industry).companyName(root.customer.account.companyName)
     .websiteUrl(root.customer.account.websiteUrl).build();
  }

  List<Contact> contacts = new ArrayList<Contact>();
  if ( root.customer != null && root.customer.contacts != null) {
   for (org.example.entity.Contact jsonContact : root.customer.contacts) {
    Contact contact = Contact.builder()
                  .email(jsonContact.email)
                  .firstName(jsonContact.firstName)
                  .lastName(jsonContact.lastName)
                  .phone(jsonContact.phone)
                  .businessTitle(jsonContact.businessTitle)
                  .build();
    contacts.add(contact);
   }
  }

  Customer customer =
Customer.builder().account(account).contacts(contacts).build();

  List<ExpectedCustomerSpend> expectedCustomerSpends = new
ArrayList<ExpectedCustomerSpend>();
  if ( root.project != null && root.project.expectedCustomerSpend != null) {
   for (org.example.entity.ExpectedCustomerSpend expectedCustomerSpendJson :
root.project.expectedCustomerSpend) {
    ExpectedCustomerSpend expectedCustomerSpend = null;
    expectedCustomerSpend = ExpectedCustomerSpend.builder()
      .amount(expectedCustomerSpendJson.amount)
      .currencyCode(expectedCustomerSpendJson.currencyCode)
```

```
      .frequency(expectedCustomerSpendJson.frequency)
      .targetCompany(expectedCustomerSpendJson.targetCompany)
      .build();
    expectedCustomerSpends.add(expectedCustomerSpend);
   }
        }

  Project project = null;
  if (root.project != null) {
   project = Project.builder().title(root.project.title)
     .customerBusinessProblem(root.project.customerBusinessProblem)

.customerUseCase(root.project.customerUseCase).deliveryModels(root.project.deliveryModels)
     .expectedCustomerSpend(expectedCustomerSpends)

.salesActivities(root.project.salesActivities).competitorName(root.project.competitorName)
     .otherSolutionDescription(root.project.otherSolutionDescription).build();
  }

  // Building the Actual CreateOpportunity Request
  UpdateOpportunityRequest updateOpportunityRequest =
UpdateOpportunityRequest.builder().catalog(root.catalog)

.identifier(root.identifier).lastModifiedDate(Instant.parse(root.lastModifiedDate))

.primaryNeedsFromAwsWithStrings(root.primaryNeedsFromAws).opportunityType(root.opportunityT
    .lifeCycle(lifeCycle)
    .customer(customer)
    .project(project)
    .partnerOpportunityIdentifier(root.partnerOpportunityIdentifier)
    .marketing(marketing)
    .nationalSecurity(root.nationalSecurity)
    .opportunityType(root.opportunityType)
    .build();

  UpdateOpportunityResponse updateResponse =
client.updateOpportunity(updateOpportunityRequest);
  System.out.println("Successfully updated opportunity: " + updateResponse);

  return updateResponse;
 } else {
  System.out.println("Opportunity cannot be updated.");
  return null;
 }
```

```
    }
}
```

- Per i dettagli sull'API, consulta la UpdateOpportunitysezione AWS SDK for Java 2.x API Reference.

Scenari

Aggiorna l'entità associata di un'opportunità

L'esempio di codice seguente mostra come:

- Dissocia una vecchia entità.
- Associa una nuova entità.

SDK per Java 2.x

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri come configurarlo ed eseguirlo nel repository Scenarios.

Aggiorna l'entità associata di un'opportunità

```
package org.example;

import static org.example.utils.Constants.*;

import org.example.utils.Constants;
import org.example.utils.ReferenceCodesUtils;

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.partnercentralselling.PartnerCentralSellingClient;
import
 software.amazon.awssdk.services.partnercentralselling.model.AssociateOpportunityRequest;
```

```
import
 software.amazon.awssdk.services.partnercentralselling.model.AssociateOpportunityResponse;
import
 software.amazon.awssdk.services.partnercentralselling.model.DisassociateOpportunityRequest;
import
 software.amazon.awssdk.services.partnercentralselling.model.DisassociateOpportunityResponse

/*
Purpose
PC-API -17 Replacing a solution
*/

public class ReplaceSolution {

 static PartnerCentralSellingClient client = PartnerCentralSellingClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(DefaultCredentialsProvider.create())
            .httpClient(ApacheHttpClient.builder().build())
            .build();

    public static void main(String[] args) {

     String opportunityId = args.length > 0 ? args[0] : OPPORTUNITY_ID;

     String entityType = "Solutions";
     String originalEntityIdentifier = "S-0000000";
     String newEntityIdentifier = "S-0011111";

     disassociateOppornititityResponse(opportunityId, entityType,
 originalEntityIdentifier );
     AssociateOpportunityResponse associateOpportunityResponse =
 associateOpportunityResponse(opportunityId, entityType, newEntityIdentifier );

     ReferenceCodesUtils.formatOutput(associateOpportunityResponse);
    }

 private static AssociateOpportunityResponse associateOpportunityResponse(String
 opportunityId, String entityType, String entityIdentifier) {

        AssociateOpportunityRequest associateOpportunityRequest =
 AssociateOpportunityRequest.builder()
    .catalog(Constants.CATALOG_TO_USE)
          .opportunityIdentifier(opportunityId)
          .relatedEntityType(entityType)
```

```
            .relatedEntityIdentifier(entityIdentifier)
            .build();

        AssociateOpportunityResponse response =
 client.associateOpportunity(associateOpportunityRequest);

        return response;
 }

 private static DisassociateOpportunityResponse
 disassociateOppornitityResponse(String opportunityId, String entityType, String
 entityIdentifier) {
  PartnerCentralSellingClient client = PartnerCentralSellingClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(DefaultCredentialsProvider.create())
            .httpClient(ApacheHttpClient.builder().build())
            .build();

        DisassociateOpportunityRequest disassociateOpportunityRequest =
 DisassociateOpportunityRequest.builder()
    .catalog(Constants.CATALOG_TO_USE)
          .opportunityIdentifier(opportunityId)
          .relatedEntityType(entityType)
          .relatedEntityIdentifier(entityIdentifier)
          .build();

        DisassociateOpportunityResponse response =
 client.disassociateOpportunity(disassociateOpportunityRequest);

        return response;
 }
}
```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella Documentazione di riferimento delle API AWS SDK for Java 2.x .
  - AssociateOpportunity
  - DisassociateOpportunity

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.

Ogni esempio include un collegamento al codice sorgente completo, 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, consulta la CreateBatchInferenceJobsezione 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 String createPersonalCompaign(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());
           return campaignResponse.campaignArn();
       } catch (PersonalizeException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
```

```
            System.exit(1);
        }
        return null;
    }
```

- Per i dettagli sull'API, consulta la CreateCampaignsezione 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, consulta la CreateDatasetsezione 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, consulta la CreateDatasetExportJobsezione 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, consulta la CreateDatasetImportJobsezione 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, consulta la CreateEventTrackersezione 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, consulta la CreateFiltersezione 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, consulta la CreateRecommendersezione 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 CreateSchemasezione 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, consulta la CreateSolutionsezione 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, consulta la CreateSolutionVersionsezione 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);
            System.out.println("Delete request sent successfully.");
```

```
        } catch (PersonalizeException e) {
            System.err.println("Error deleting campaign: " +
 e.awsErrorDetails().errorMessage());
            throw new RuntimeException(e);
        }
    }
}
```

- Per i dettagli sull'API, consulta la DeleteCampaignsezione 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().statusCod

            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 DeleteEventTrackersezione 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 DeleteSolutionsezione 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 DescribeCampaignsezione 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 DescribeRecipesezione 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 DescribeSolutionsezione 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 ListCampaignssezione 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 ListDatasetGroupssezione 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 ListSolutionssezione 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 UpdateCampaignsezione 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.

Ogni esempio include un collegamento al codice sorgente completo, 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 PutEventssezione 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 PutUserssezione 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.

Ogni esempio include un collegamento al codice sorgente completo, 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 GetPersonalizedRankingsezione 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.xAPI 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.

Ogni esempio include un collegamento al codice sorgente completo, 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 CreateAppsezione 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 CreateCampaignsezione 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 CreateExportJobsezione 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 CreateImportJobsezione 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 CreateSegmentsezione 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 DeleteAppsezione 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();

        deletePinEncpoint(pinpoint, appId, endpointId);
        pinpoint.close();
    }

    public static void deletePinEncpoint(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 DeleteEndpointsezione 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 GetEndpointsezione 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("Segement " + 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 GetSegmentssezione 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 GetSmsChannelsezione 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 GetUserEndpointssezione 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 SendEmailMessage {

        // The character encoding the you want to use for the subject line and
        // message body of the email.
        public static String charset = "UTF-8";

    // The body of the email for recipients whose email clients support HTML
 content.
    static final String body = """
        Amazon Pinpoint test (AWS SDK for Java 2.x)

        This email was sent through the Amazon Pinpoint Email API using the AWS SDK
 for Java 2.x

        """;

        public static void main(String[] args) {
                final String usage = """
```

```
                                Usage:    <subject> <appId> <senderAddress>
<toAddress>

           Where:
              subject - The email subject to use.
              senderAddress - The from address. This address has to be verified in
Amazon Pinpoint in the region you're using to send email\s
              toAddress - The to address. This address has to be verified in Amazon
Pinpoint in the region you're using to send email\s
           """;

       if (args.length != 3) {
           System.out.println(usage);
           System.exit(1);
       }

       String subject = args[0];
       String senderAddress = args[1];
       String toAddress = args[2];
       System.out.println("Sending a message");
       PinpointEmailClient pinpoint = PinpointEmailClient.builder()
           .region(Region.US_EAST_1)
           .build();

       sendEmail(pinpoint, subject, senderAddress, toAddress);
       System.out.println("Email was sent");
       pinpoint.close();
   }

   public static void sendEmail(PinpointEmailClient pinpointEmailClient, String
subject, String senderAddress, String toAddress) {
       try {
           Content content = Content.builder()
               .data(body)
               .build();

           Body messageBody = Body.builder()
               .text(content)
               .build();

           Message message = Message.builder()
               .body(messageBody)
               .subject(Content.builder().data(subject).build())
               .build();
```

```
            Destination destination = Destination.builder()
                .toAddresses(toAddress)
                .build();

            EmailContent emailContent = EmailContent.builder()
                .simple(message)
                .build();

            SendEmailRequest sendEmailRequest = SendEmailRequest.builder()
                .fromEmailAddress(senderAddress)
                .destination(destination)
                .content(emailContent)
                .build();

            pinpointEmailClient.sendEmail(sendEmailRequest);
            System.out.println("Message Sent");

        } catch (PinpointException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

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.sendMessages(request);
                        MessageResponse msg1 = response.messageResponse();
                        Map map1 = msg1.result();

                        // Write out the result of sendMessage.
                        map1.forEach((k, v) -> System.out.println((k + ":" + v)));

                } catch (PinpointException e) {
                        System.err.println(e.awsErrorDetails().errorMessage());
                        System.exit(1);
                }
        }
```

```
}
```

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.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.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class SendMessageBatch {

    // The type of SMS message that you want to send. If you plan to send
    // time-sensitive content, specify TRANSACTIONAL. If you plan to send
    // marketing-related content, specify PROMOTIONAL.
    public static String messageType = "TRANSACTIONAL";

    // The registered keyword associated with the originating short code.
    public static String registeredKeyword = "myKeyword";

    // The sender ID to use when sending the message. Support for sender ID
    // varies by country or region. For more information, see
    // https://docs.aws.amazon.com/pinpoint/latest/userguide/channels-sms-
countries.html
```

```
   public static String senderId = "MySenderID";

   public static void main(String[] args) {
       final String usage = """

               Usage:   <message> <appId> <originationNumber> <destinationNumber>
<destinationNumber1>\s

               Where:
                 message - The body of the message to send.
                 appId - The Amazon Pinpoint project/application ID to use when you
send this message.
                 originationNumber - The phone number or short code that you
specify has to be associated with your Amazon Pinpoint account. For best results,
specify long codes in E.164 format (for example, +1-555-555-5654).
                 destinationNumber - The recipient's phone number.  For best
results, you should specify the phone number in E.164 format (for example,
+1-555-555-5654).
                 destinationNumber1 - The second recipient's phone number.  For
best results, you should specify the phone number in E.164 format (for example,
+1-555-555-5654).\s
               """;

       if (args.length != 5) {
           System.out.println(usage);
           System.exit(1);
       }

       String message = args[0];
       String appId = args[1];
       String originationNumber = args[2];
       String destinationNumber = args[3];
       String destinationNumber1 = args[4];
       System.out.println("Sending a message");
       PinpointClient pinpoint = PinpointClient.builder()
               .region(Region.US_EAST_1)
               .build();

       sendSMSMessage(pinpoint, message, appId, originationNumber,
destinationNumber, destinationNumber1);
       pinpoint.close();
   }
```

```
   public static void sendSMSMessage(PinpointClient pinpoint, String message,
String appId,
                                     String originationNumber,
                                     String destinationNumber, String
destinationNumber1) {
       try {
           Map<String, AddressConfiguration> addressMap = new HashMap<String,
AddressConfiguration>();
           AddressConfiguration addConfig = AddressConfiguration.builder()
                   .channelType(ChannelType.SMS)
                   .build();

           // Add an entry to the Map object for each number to whom you want to
send a
           // message.
           addressMap.put(destinationNumber, addConfig);
           addressMap.put(destinationNumber1, addConfig);
           SMSMessage smsMessage = SMSMessage.builder()
                   .body(message)
                   .messageType(messageType)
                   .originationNumber(originationNumber)
                   .senderId(senderId)
                   .keyword(registeredKeyword)
                   .build();

           // Create a DirectMessageConfiguration object.
           DirectMessageConfiguration direct = DirectMessageConfiguration.builder()
                   .smsMessage(smsMessage)
                   .build();

           MessageRequest msgReq = MessageRequest.builder()
                   .addresses(addressMap)
                   .messageConfiguration(direct)
                   .build();

           // Create a SendMessagesRequest object.
           SendMessagesRequest request = SendMessagesRequest.builder()
                   .applicationId(appId)
                   .messageRequest(msgReq)
                   .build();

           SendMessagesResponse response = pinpoint.sendMessages(request);
           MessageResponse msg1 = response.messageResponse();
           Map map1 = msg1.result();
```

```
            // Write out the result of sendMessage.
            map1.forEach((k, v) -> System.out.println((k + ":" + v)));

        } catch (PinpointException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Per i dettagli sull'API, consulta la SendMessagessezione 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 UpdateEndpointsezione 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.

Ogni esempio include un collegamento al codice sorgente completo, 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.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * 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 SendVoiceMessagesezione 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.

Gli scenari sono esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio o combinate con altri Servizi AWS.

Ogni esempio include un collegamento al codice sorgente completo, dove puoi trovare istruzioni su come configurare ed eseguire il codice nel contesto.

Argomenti

- Azioni
- Scenari

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 DescribeVoicessezione 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 ListLexiconssezione 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 SynthesizeSpeechsezione AWS SDK for Java 2.x API Reference.

Scenari

Crea un'applicazione per analizzare il feedback dei clienti

L'esempio di codice seguente mostra come creare un'applicazione che analizza le schede dei commenti dei clienti, le traduce dalla loro lingua originale, ne determina il sentiment e genera un file audio dal testo tradotto.

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

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 nozioni di base sono esempi di codice che mostrano come eseguire le operazioni essenziali all'interno di un servizio.

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.

Gli scenari sono esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio o combinate con altri Servizi AWS.

Ogni esempio include un collegamento al codice sorgente completo, 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 i dettagli sull'API, consulta Descrivi DBInstances in AWS SDK for Java 2.x API Reference.

Argomenti

- Nozioni di base
- Azioni

- Scenari
- Esempi serverless

Nozioni di base

Informazioni di base

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 altro su 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)
            .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 userName,
            String userPassword) {

        try {
            CreateDbInstanceRequest instanceRequest =
CreateDbInstanceRequest.builder()
                .dbInstanceIdentifier(dbInstanceIdentifier)
                .allocatedStorage(100)
                .dbName(dbName)
                .engine("mysql")
                .dbInstanceClass("db.t3.medium") // Updated to a supported class
```

```
                .engineVersion("8.0.32")         // Updated to a supported version
                .storageType("gp2")             // Changed to General Purpose SSD
(gp2)
                .masterUsername(userName)
                .masterUserPassword(userPassword)
                .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 .
  - CreaDBInstance
  - Crea DBParameter gruppo
  - CreaDBSnapshot
  - EliminaDBInstance
  - Elimina DBParameter gruppo
  - Descrivi DBEngine versioni
  - Descriva DBInstances
  - DBParameterDescrivi gruppi
  - Descriva DBParameters
  - Descriva DBSnapshots
  - DescribeOrderableDBInstanceOpzioni
  - Modifica DBParameter gruppo

Azioni

`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.

```
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.t3.medium") // Updated to a supported class
                .engineVersion("8.0.32")         // Updated to a supported version
                .storageType("gp2")             // Changed to General Purpose SSD
(gp2)
                .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 i dettagli sull'API, consulta Create DBInstance in AWS SDK for Java 2.x API Reference.

`CreateDBParameterGroup`

Il seguente esempio di codice mostra come utilizzare`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 Create DBParameter Group 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 i dettagli sull'API, consulta Create DBSnapshot 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.

```
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 i dettagli sull'API, consulta Delete DBInstance in AWS SDK for Java 2.x API Reference.

`DeleteDBParameterGroup`

Il seguente esempio di codice mostra come utilizzare`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 Delete DBParameter Group 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 DescribeAccountAttributessezione 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 Descrivi DBEngine le versioni 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 i dettagli sull'API, consulta Descrivi DBInstances in AWS SDK for Java 2.x API Reference.

`DescribeDBParameterGroups`

Il seguente esempio di codice mostra come utilizzare`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 Descrivere DBParameter i gruppi 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 i dettagli sull'API, consulta Descrivi DBParameters 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 RdsUtilitiesclasse 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 Generate RDSAuth Token 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 i dettagli sull'API, consulta Modify DBInstance in AWS SDK for Java 2.x API Reference.

`ModifyDBParameterGroup`

Il seguente esempio di codice mostra come utilizzare`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 Modify DBParameter Group 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 i dettagli sull'API, consulta Reboot DBInstance in AWS SDK for Java 2.x API Reference.

Scenari

Creazione di un tracciatore di elementi di lavoro di Aurora Serverless

Il seguente esempio di codice mostra come creare un'applicazione Web che tiene traccia degli elementi di lavoro in un database Amazon Aurora Serverless e utilizza Amazon Simple Email Service (Amazon SES) per inviare report.

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

Esempi serverless

Connessione a un database Amazon RDS in una funzione Lambda

Il seguente esempio di codice mostra come implementare una funzione Lambda che si connette a un database RDS. La funzione effettua una semplice richiesta al database e restituisce il risultato.

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.

Connessione a un database Amazon RDS in una funzione Lambda tramite Java.

```
import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.APIGatewayProxyRequestEvent;
import com.amazonaws.services.lambda.runtime.events.APIGatewayProxyResponseEvent;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
```

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rdsdata.RdsDataClient;
import software.amazon.awssdk.services.rdsdata.model.ExecuteStatementRequest;
import software.amazon.awssdk.services.rdsdata.model.ExecuteStatementResponse;
import software.amazon.awssdk.services.rdsdata.model.Field;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;

public class RdsLambdaHandler implements RequestHandler<APIGatewayProxyRequestEvent,
 APIGatewayProxyResponseEvent> {

    @Override
    public APIGatewayProxyResponseEvent handleRequest(APIGatewayProxyRequestEvent
 event, Context context) {
        APIGatewayProxyResponseEvent response = new APIGatewayProxyResponseEvent();

        try {
            // Obtain auth token
            String token = createAuthToken();

            // Define connection configuration
            String connectionString = String.format("jdbc:mysql://%s:%s/%s?
useSSL=true&requireSSL=true",
                    System.getenv("ProxyHostName"),
                    System.getenv("Port"),
                    System.getenv("DBName"));

            // Establish a connection to the database
            try (Connection connection =
 DriverManager.getConnection(connectionString, System.getenv("DBUserName"), token);
                 PreparedStatement statement = connection.prepareStatement("SELECT ?
 + ? AS sum")) {

                statement.setInt(1, 3);
                statement.setInt(2, 2);

                try (ResultSet resultSet = statement.executeQuery()) {
                    if (resultSet.next()) {
                        int sum = resultSet.getInt("sum");
                        response.setStatusCode(200);
                        response.setBody("The selected sum is: " + sum);
```

```
                    }
                }
            }

        } catch (Exception e) {
            response.setStatusCode(500);
            response.setBody("Error: " + e.getMessage());
        }

        return response;
    }

    private String createAuthToken() {
        // Create RDS Data Service client
        RdsDataClient rdsDataClient = RdsDataClient.builder()
                .region(Region.of(System.getenv("AWS_REGION")))
                .credentialsProvider(DefaultCredentialsProvider.create())
                .build();

        // Define authentication request
        ExecuteStatementRequest request = ExecuteStatementRequest.builder()
                .resourceArn(System.getenv("ProxyHostName"))
                .secretArn(System.getenv("DBUserName"))
                .database(System.getenv("DBName"))
                .sql("SELECT 'RDS IAM Authentication'")
                .build();

        // Execute request and obtain authentication token
        ExecuteStatementResponse response = rdsDataClient.executeStatement(request);
        Field tokenField = response.records().get(0).get(0);

        return tokenField.stringValue();
    }
}
```

Esempi di Amazon RDS Data Service 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 RDS Data Service.

Gli scenari sono esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio o combinate con altri Servizi AWS.

Ogni esempio include un collegamento al codice sorgente completo, dove puoi trovare istruzioni su come configurare ed eseguire il codice nel contesto.

Argomenti

- Scenari

Scenari

Creazione di un tracciatore di elementi di lavoro di Aurora Serverless

Il seguente esempio di codice mostra come creare un'applicazione Web che tiene traccia degli elementi di lavoro in un database Amazon Aurora Serverless e utilizza Amazon Simple Email Service (Amazon SES) per inviare report.

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

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 nozioni di base sono esempi di codice che mostrano come eseguire le operazioni essenziali all'interno di un servizio.

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.

Gli scenari sono esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio o combinate con altri Servizi AWS.

Ogni esempio include un collegamento al codice sorgente completo, 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 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.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 la DescribeClusterssezione AWS SDK for Java 2.x API Reference.

Argomenti

- Nozioni di base
- Azioni
- Scenari

Nozioni di base

Informazioni di base

L'esempio di codice seguente mostra come:

- Crea un cluster Redshift.
- Elenca i database nel cluster.

- Crea una tabella denominata Movies.
- Compilate la tabella Film.
- Eseguite una query nella tabella Film per anno.
- Modifica il cluster Redshift.
- Elimina il cluster Amazon Redshift.

SDK per Java 2.x

> **Note**
>
> C'è altro da fare. 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 Amazon Redshift.

```
import com.example.redshift.User;
import com.google.gson.Gson;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.redshift.model.ClusterAlreadyExistsException;
import software.amazon.awssdk.services.redshift.model.CreateClusterResponse;
import software.amazon.awssdk.services.redshift.model.DeleteClusterResponse;
import software.amazon.awssdk.services.redshift.model.ModifyClusterResponse;
import software.amazon.awssdk.services.redshift.model.RedshiftException;
import software.amazon.awssdk.services.redshiftdata.model.ExecuteStatementResponse;
import software.amazon.awssdk.services.redshiftdata.model.RedshiftDataException;
import java.util.Scanner;
import java.util.concurrent.CompletableFuture;
import software.amazon.awssdk.services.secretsmanager.SecretsManagerClient;
import software.amazon.awssdk.services.secretsmanager.model.GetSecretValueRequest;
import software.amazon.awssdk.services.secretsmanager.model.GetSecretValueResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including 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 that specifies user name and password, 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. 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", "-");
    private static final Logger logger =
 LoggerFactory.getLogger(RedshiftScenario.class);

    static RedshiftActions redshiftActions = new RedshiftActions();
    public static void main(String[] args) throws Exception {
        final String usage = """

            Usage:
                <jsonFilePath> <secretName>\s

            Where:
                jsonFilePath - The path to the Movies JSON file (you can locate that
 file in ../../../resources/sample_files/movies.json)
                secretName - The name of the secret that belongs to Secret Manager
 that stores the user name and password used in this scenario.
```

```
            """;

        if (args.length != 2) {
            logger.info(usage);
            return;
        }

        String jsonFilePath = args[0];
        String secretName = args[1];
        Scanner scanner = new Scanner(System.in);
        logger.info(DASHES);
        logger.info("Welcome to the Amazon Redshift SDK Basics scenario.");
        logger.info("""
            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.
            """);

        logger.info("Lets get started...");
        logger.info("""
            First, we will retrieve the user name and password from Secrets Manager.

            Using Amazon Secrets Manager to store Redshift credentials provides
 several security benefits.
            It allows you to securely store and manage sensitive information, such
 as passwords, API keys, and
            database credentials, without embedding them directly in your
 application code.

            More information can be found here:

            https://docs.aws.amazon.com/secretsmanager/latest/userguide/
integrating_how-services-use-secrets_RS.html
            """);
```

```
        Gson gson = new Gson();
        User user = gson.fromJson(String.valueOf(getSecretValues(secretName)),
User.class);
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        try {
            runScenario(user, scanner, jsonFilePath);
        } catch (RuntimeException e) {
            e.printStackTrace();
        } catch (Throwable e) {
            throw new RuntimeException(e);
        }
    }

    private static void runScenario(User user, Scanner scanner,  String
jsonFilePath) throws Throwable {
        String databaseName = "dev";
        System.out.println(DASHES);
        logger.info("Create a Redshift Cluster");
        logger.info("A Redshift cluster refers to the collection of computing
resources and storage that work together to process and analyze large volumes of
data.");
        logger.info("Enter a cluster id value or accept the default by hitting Enter
(default is redshift-cluster-movies): ");
        String userClusterId = scanner.nextLine();
        String clusterId = userClusterId.isEmpty() ? "redshift-cluster-movies" :
userClusterId;
        try {
            CompletableFuture<CreateClusterResponse> future =
redshiftActions.createClusterAsync(clusterId, user.getUserName(),
user.getUserPassword());
            CreateClusterResponse response = future.join();
            logger.info("Cluster successfully created. Cluster Identifier {} ",
response.cluster().clusterIdentifier());

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof ClusterAlreadyExistsException) {
                logger.info("The Cluster {} already exists. Moving on...",
clusterId);
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
```

```
        }
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("Wait until {} is available.", clusterId);
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Void> future =
 redshiftActions.waitForClusterReadyAsync(clusterId);
            future.join();
            logger.info("Cluster is ready!");

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof RedshiftException redshiftEx) {
                logger.info("Redshift error occurred: Error message: {}, Error code
 {}", redshiftEx.getMessage(), redshiftEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            throw cause;
        }
        logger.info(DASHES);

        logger.info(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);

        logger.info(databaseInfo);
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("List databases in {} ",clusterId);
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Void> future =
 redshiftActions.listAllDatabasesAsync(clusterId, user.getUserName(), "dev");
```

```
            future.join();
            logger.info("Databases listed successfully.");

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof RedshiftDataException redshiftEx) {
                logger.error("Redshift Data error occurred: {} Error code: {}",
redshiftEx.getMessage(), redshiftEx.awsErrorDetails().errorCode());
            } else {
                logger.error("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("Now you will create a table named Movies.");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<ExecuteStatementResponse> future =
redshiftActions.createTableAsync(clusterId, databaseName, user.getUserName());
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof RedshiftDataException redshiftEx) {
                logger.info("Redshift Data error occurred: {} Error code: {}",
redshiftEx.getMessage(), redshiftEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("Populate the Movies table using the Movies.json file.");
        logger.info("Specify the number of records you would like to add to the
Movies Table.");
        logger.info("Please enter a value between 50 and 200.");
        int numRecords;
        do {
            logger.info("Enter a value: ");
            while (!scanner.hasNextInt()) {
```

```
                logger.info("Invalid input. Please enter a value between 50 and
200.");
                logger.info("Enter a year: ");
                scanner.next();
            }
            numRecords = scanner.nextInt();
        } while (numRecords < 50 || numRecords > 200);
        try {
            redshiftActions.popTableAsync(clusterId, databaseName,
user.getUserName(), jsonFilePath, numRecords).join();  // Wait for the operation to
complete
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof RedshiftDataException redshiftEx) {
                logger.info("Redshift Data error occurred: {} Error code: {}",
redshiftEx.getMessage(), redshiftEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("Query the Movies table by year. Enter a value between
2012-2014.");
        int movieYear;
        do {
            logger.info("Enter a year: ");
            while (!scanner.hasNextInt()) {
                logger.info("Invalid input. Please enter a valid year between 2012
and 2014.");
                logger.info("Enter a year: ");
                scanner.next();
            }
            movieYear = scanner.nextInt();
            scanner.nextLine();
        } while (movieYear < 2012 || movieYear > 2014);

        String id;
        try {
```

```
            CompletableFuture<String> future =
redshiftActions.queryMoviesByYearAsync(databaseName, user.getUserName(), movieYear,
clusterId);
            id = future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof RedshiftDataException redshiftEx) {
                logger.info("Redshift Data error occurred: {} Error code: {}",
redshiftEx.getMessage(), redshiftEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }

        logger.info("The identifier of the statement is " + id);
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Void> future =
redshiftActions.checkStatementAsync(id);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof RedshiftDataException redshiftEx) {
                logger.info("Redshift Data error occurred: {} Error code: {}",
redshiftEx.getMessage(), redshiftEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Void> future = redshiftActions.getResultsAsync(id);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof RedshiftDataException redshiftEx) {
                logger.info("Redshift Data error occurred: {} Error code: {}",
redshiftEx.getMessage(), redshiftEx.awsErrorDetails().errorCode());
            } else {
```

```
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("Now you will modify the Redshift cluster.");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<ModifyClusterResponse> future =
redshiftActions.modifyClusterAsync(clusterId);;
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof RedshiftDataException redshiftEx) {
                logger.info("Redshift Data error occurred: {} Error code: {}",
redshiftEx.getMessage(), redshiftEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("Would you like to delete the Amazon Redshift cluster? (y/n)");
        String delAns = scanner.nextLine().trim();
        if (delAns.equalsIgnoreCase("y")) {
            logger.info("You selected to delete {} ", clusterId);
            waitForInputToContinue(scanner);
            try {
                CompletableFuture<DeleteClusterResponse> future =
redshiftActions.deleteRedshiftClusterAsync(clusterId);;
                future.join();

            } catch (RuntimeException rt) {
                Throwable cause = rt.getCause();
                if (cause instanceof RedshiftDataException redshiftEx) {
                    logger.info("Redshift Data error occurred: {} Error code: {}",
redshiftEx.getMessage(), redshiftEx.awsErrorDetails().errorCode());
```

```
                } else {
                    logger.info("An unexpected error occurred: {}",
rt.getMessage());
                }
                throw cause;
            }
        } else {
            logger.info("The {}  was not deleted", clusterId);
        }
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("This concludes the Amazon Redshift SDK Basics scenario.");
        logger.info(DASHES);
    }

    private static SecretsManagerClient getSecretClient() {
        Region region = Region.US_EAST_1;
        return SecretsManagerClient.builder()
            .region(region)
            .build();
    }

    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.");
            }
        }
    }

    // Get the Amazon Redshift credentials from AWS Secrets Manager.
    private static String getSecretValues(String secretName) {
        SecretsManagerClient secretClient = getSecretClient();
        GetSecretValueRequest valueRequest = GetSecretValueRequest.builder()
```

```
            .secretId(secretName)
            .build();

        GetSecretValueResponse valueResponse =
 secretClient.getSecretValue(valueRequest);
        return valueResponse.secretString();
    }
}
```

Una classe wrapper per i metodi SDK di Amazon Redshift.

```
public class RedshiftActions {

    private static final Logger logger =
 LoggerFactory.getLogger(RedshiftActions.class);
    private static RedshiftDataAsyncClient redshiftDataAsyncClient;

    private static RedshiftAsyncClient redshiftAsyncClient;

    private static RedshiftAsyncClient getAsyncClient() {
        if (redshiftAsyncClient == null) {
            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                .maxConcurrency(100)
                .connectionTimeout(Duration.ofSeconds(60))
                .readTimeout(Duration.ofSeconds(60))
                .writeTimeout(Duration.ofSeconds(60))
                .build();

            ClientOverrideConfiguration overrideConfig =
 ClientOverrideConfiguration.builder()
                .apiCallTimeout(Duration.ofMinutes(2))
                .apiCallAttemptTimeout(Duration.ofSeconds(90))
                .retryStrategy(RetryMode.STANDARD)
                .build();

            redshiftAsyncClient = RedshiftAsyncClient.builder()
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return redshiftAsyncClient;
    }
```

```
   private static RedshiftDataAsyncClient getAsyncDataClient() {
       if (redshiftDataAsyncClient == null) {
           SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
               .maxConcurrency(100)
               .connectionTimeout(Duration.ofSeconds(60))
               .readTimeout(Duration.ofSeconds(60))
               .writeTimeout(Duration.ofSeconds(60))
               .build();

           ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
               .apiCallTimeout(Duration.ofMinutes(2))
               .apiCallAttemptTimeout(Duration.ofSeconds(90))
               .retryStrategy(RetryMode.STANDARD)
               .build();

           redshiftDataAsyncClient = RedshiftDataAsyncClient.builder()
               .httpClient(httpClient)
               .overrideConfiguration(overrideConfig)
               .build();
       }
       return redshiftDataAsyncClient;
   }

   /**
    * Creates a new Amazon Redshift cluster asynchronously.
    * @param clusterId     the unique identifier for the cluster
    * @param username      the username for the administrative user
    * @param userPassword  the password for the administrative user
    * @return a CompletableFuture that represents the asynchronous operation of
creating the cluster
    * @throws RuntimeException if the cluster creation fails
    */
   public CompletableFuture<CreateClusterResponse> createClusterAsync(String
clusterId, String username, String userPassword) {
       CreateClusterRequest clusterRequest = CreateClusterRequest.builder()
           .clusterIdentifier(clusterId)
           .masterUsername(username)
           .masterUserPassword(userPassword)
           .nodeType("ra3.4xlarge")
           .publiclyAccessible(true)
           .numberOfNodes(2)
           .build();
```

```
        return getAsyncClient().createCluster(clusterRequest)
            .whenComplete((response, exception) -> {
                if (response != null) {
                    logger.info("Created cluster ");
                } else {
                    throw new RuntimeException("Failed to create cluster: " +
exception.getMessage(), exception);
                }
            });
    }

    /**
     * Waits asynchronously for the specified cluster to become available.
     * @param clusterId the identifier of the cluster to wait for
     * @return a {@link CompletableFuture} that completes when the cluster is ready
     */
    public CompletableFuture<Void> waitForClusterReadyAsync(String clusterId) {
        DescribeClustersRequest clustersRequest = DescribeClustersRequest.builder()
            .clusterIdentifier(clusterId)
            .build();

        logger.info("Waiting for cluster to become available. This may take a few
minutes.");
        long startTime = System.currentTimeMillis();

        // Recursive method to poll the cluster status.
        return checkClusterStatusAsync(clustersRequest, startTime);
    }

    private CompletableFuture<Void> checkClusterStatusAsync(DescribeClustersRequest
clustersRequest, long startTime) {
        return getAsyncClient().describeClusters(clustersRequest)
            .thenCompose(clusterResponse -> {
                List<Cluster> clusterList = clusterResponse.clusters();
                boolean clusterReady = false;
                for (Cluster cluster : clusterList) {
                    if ("available".equals(cluster.clusterStatus())) {
                        clusterReady = true;
                        break;
                    }
                }

                if (clusterReady) {
```

```
                    logger.info(String.format("Cluster is available!"));
                    return CompletableFuture.completedFuture(null);
                } else {
                    long elapsedTimeMillis = System.currentTimeMillis() - startTime;
                    long elapsedSeconds = elapsedTimeMillis / 1000;
                    long minutes = elapsedSeconds / 60;
                    long seconds = elapsedSeconds % 60;
                    System.out.printf("\rElapsed Time: %02d:%02d - Waiting for
cluster...", minutes, seconds);
                    System.out.flush();

                    // Wait 1 second before the next status check
                    return CompletableFuture.runAsync(() -> {
                        try {
                            TimeUnit.SECONDS.sleep(1);
                        } catch (InterruptedException e) {
                            throw new RuntimeException("Error during sleep: " +
e.getMessage(), e);
                        }
                    }).thenCompose(ignored ->
checkClusterStatusAsync(clustersRequest, startTime));
                }
            }).exceptionally(exception -> {
                throw new RuntimeException("Failed to get cluster status: " +
exception.getMessage(), exception);
            });
    }

    /**
     * Lists all databases asynchronously for the specified cluster, database user,
and database.
     * @param clusterId the identifier of the cluster to list databases for
     * @param dbUser the database user to use for the list databases request
     * @param database the database to list databases for
     * @return a {@link CompletableFuture} that completes when the database listing
is complete, or throws a {@link RuntimeException} if there was an error
     */
    public CompletableFuture<Void> listAllDatabasesAsync(String clusterId, String
dbUser, String database) {
        ListDatabasesRequest databasesRequest = ListDatabasesRequest.builder()
            .clusterIdentifier(clusterId)
            .dbUser(dbUser)
            .database(database)
            .build();
```

```
        // Asynchronous paginator for listing databases.
        ListDatabasesPublisher databasesPaginator =
getAsyncDataClient().listDatabasesPaginator(databasesRequest);
        CompletableFuture<Void> future = databasesPaginator.subscribe(response -> {
            response.databases().forEach(db -> {
                logger.info("The database name is {} ", db);
            });
        });

        // Return the future for asynchronous handling.
        return future.exceptionally(exception -> {
            throw new RuntimeException("Failed to list databases: " +
exception.getMessage(), exception);
        });
    }

    /**
     * Creates an asynchronous task to execute a SQL statement for creating a new
table.
     *
     * @param clusterId    the identifier of the Amazon Redshift cluster
     * @param databaseName the name of the database to create the table in
     * @param userName     the username to use for the database connection
     * @return a {@link CompletableFuture} that completes with the result of the SQL
statement execution
     * @throws RuntimeException if there is an error creating the table
     */
    public CompletableFuture<ExecuteStatementResponse> createTableAsync(String
clusterId, String databaseName, String userName) {
        ExecuteStatementRequest createTableRequest =
ExecuteStatementRequest.builder()
            .clusterIdentifier(clusterId)
            .dbUser(userName)
            .database(databaseName)
            .sql("CREATE TABLE Movies (" +
                "id INT PRIMARY KEY, " +
                "title VARCHAR(100), " +
                "year INT)")
            .build();

        return getAsyncDataClient().executeStatement(createTableRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
```

```
                    throw new RuntimeException("Error creating table: " +
exception.getMessage(), exception);
                } else {
                    logger.info("Table created: Movies");
                }
            });
    }

    /**
     * Asynchronously pops a table from a JSON file.
     *
     * @param clusterId   the ID of the cluster
     * @param databaseName the name of the database
     * @param userName    the username
     * @param fileName    the name of the JSON file
     * @param number      the number of records to process
     * @return a CompletableFuture that completes with the number of records added
to the Movies table
     */
    public CompletableFuture<Integer> popTableAsync(String clusterId, String
databaseName, String userName, String fileName, int number) {
        return CompletableFuture.supplyAsync(() -> {
                try {
                    JsonParser parser = new JsonFactory().createParser(new
File(fileName));
                    JsonNode rootNode = new ObjectMapper().readTree(parser);
                    Iterator<JsonNode> iter = rootNode.iterator();
                    return iter;
                } catch (IOException e) {
                    throw new RuntimeException("Failed to read or parse JSON file: "
+ e.getMessage(), e);
                }
            }).thenCompose(iter -> processNodesAsync(clusterId, databaseName,
userName, iter, number))
            .whenComplete((result, exception) -> {
                if (exception != null) {
                    logger.info("Error {} ", exception.getMessage());
                } else {
                    logger.info("{} records were added to the Movies table." ,
result);
                }
            });
    }
```

```
   private CompletableFuture<Integer> processNodesAsync(String clusterId, String
databaseName, String userName, Iterator<JsonNode> iter, int number) {
       return CompletableFuture.supplyAsync(() -> {
           int t = 0;
           try {
               while (iter.hasNext()) {
                   if (t == number)
                       break;
                   JsonNode currentNode = 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);";
                   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);

                   ExecuteStatementRequest insertStatementRequest =
ExecuteStatementRequest.builder()
                       .clusterIdentifier(clusterId)
                       .sql(sqlStatement)
                       .database(databaseName)
                       .dbUser(userName)
                       .parameters(parameterList)
                       .build();

                   getAsyncDataClient().executeStatement(insertStatementRequest);
```

```
                    logger.info("Inserted: " + title + " (" + year + ")");
                    t++;
                }
            } catch (RedshiftDataException e) {
                throw new RuntimeException("Error inserting data: " +
e.getMessage(), e);
            }
            return t;
        });
    }

    /**
     * Checks the status of an SQL statement asynchronously and handles the
completion of the statement.
     *
     * @param sqlId the ID of the SQL statement to check
     * @return a {@link CompletableFuture} that completes when the SQL statement's
status is either "FINISHED" or "FAILED"
     */
    public CompletableFuture<Void> checkStatementAsync(String sqlId) {
        DescribeStatementRequest statementRequest =
DescribeStatementRequest.builder()
            .id(sqlId)
            .build();

        return getAsyncDataClient().describeStatement(statementRequest)
            .thenCompose(response -> {
                String status = response.statusAsString();
                logger.info("... Status: {} ", status);

                if ("FAILED".equals(status)) {
                    throw new RuntimeException("The Query Failed. Ending program");
                } else if ("FINISHED".equals(status)) {
                    return CompletableFuture.completedFuture(null);
                } else {
                    // Sleep for 1 second and recheck status
                    return CompletableFuture.runAsync(() -> {
                        try {
                            TimeUnit.SECONDS.sleep(1);
                        } catch (InterruptedException e) {
                            throw new RuntimeException("Error during sleep: " +
e.getMessage(), e);
                        }
```

```
                    }).thenCompose(ignore -> checkStatementAsync(sqlId)); //
Recursively call until status is FINISHED or FAILED
                }
            }).whenComplete((result, exception) -> {
                if (exception != null) {
                    // Handle exceptions
                    logger.info("Error: {} ", exception.getMessage());
                } else {
                    logger.info("The statement is finished!");
                }
            });
    }

    /**
     * Asynchronously retrieves the results of a statement execution.
     *
     * @param statementId the ID of the statement for which to retrieve the results
     * @return a {@link CompletableFuture} that completes when the statement result
has been processed
     */
    public CompletableFuture<Void> getResultsAsync(String statementId) {
        GetStatementResultRequest resultRequest =
GetStatementResultRequest.builder()
            .id(statementId)
            .build();

        return getAsyncDataClient().getStatementResult(resultRequest)
            .handle((response, exception) -> {
                if (exception != null) {
                    logger.info("Error getting statement result {} ",
exception.getMessage());
                    throw new RuntimeException("Error getting statement result: " +
exception.getMessage(), exception);
                }

                // Extract and print the field values using streams if the response
is valid.
                response.records().stream()
                    .flatMap(List::stream)
                    .map(Field::stringValue)
                    .filter(value -> value != null)
                    .forEach(value -> System.out.println("The Movie title field is "
+ value));
```

```
                return response;
            }).thenAccept(response -> {
                // Optionally add more logic here if needed after handling the
response
            });
    }


    /**
     * Asynchronously queries movies by a given year from a Redshift database.
     *
     * @param database    the name of the database to query
     * @param dbUser      the user to connect to the database with
     * @param year        the year to filter the movies by
     * @param clusterId   the identifier of the Redshift cluster to connect to
     * @return a {@link CompletableFuture} containing the response ID of the
executed SQL statement
     */
    public CompletableFuture<String> queryMoviesByYearAsync(String database,
                                                             String dbUser,
                                                             int year,
                                                             String clusterId)
{

        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();

        return CompletableFuture.supplyAsync(() -> {
            try {
                ExecuteStatementResponse response =
getAsyncDataClient().executeStatement(statementRequest).join(); // Use join() to
wait for the result
                return response.id();
```

```
            } catch (RedshiftDataException e) {
                throw new RuntimeException("Error executing statement: " +
e.getMessage(), e);
            }
        }).exceptionally(exception -> {
            logger.info("Error: {}", exception.getMessage());
            return "";
        });
    }

    /**
     * Modifies an Amazon Redshift cluster asynchronously.
     *
     * @param clusterId the identifier of the cluster to be modified
     * @return a {@link CompletableFuture} that completes when the cluster
modification is complete
     */
    public CompletableFuture<ModifyClusterResponse> modifyClusterAsync(String
clusterId) {
        ModifyClusterRequest modifyClusterRequest = ModifyClusterRequest.builder()
            .clusterIdentifier(clusterId)
            .preferredMaintenanceWindow("wed:07:30-wed:08:00")
            .build();

        return getAsyncClient().modifyCluster(modifyClusterRequest)
            .whenComplete((clusterResponse, exception) -> {
                if (exception != null) {
                    if (exception.getCause() instanceof RedshiftException) {
                        logger.info("Error: {} ", exception.getMessage());
                    } else {
                        logger.info("Unexpected error: {} ",
exception.getMessage());
                    }
                } else {
                    logger.info("The modified cluster was successfully modified and
has "
                        + clusterResponse.cluster().preferredMaintenanceWindow() + "
as the maintenance window");
                }
            });
    }

    /**
     * Deletes a Redshift cluster asynchronously.
```

```
     *
     * @param clusterId the identifier of the Redshift cluster to be deleted
     * @return a {@link CompletableFuture} that represents the asynchronous
 operation of deleting the Redshift cluster
     */
    public CompletableFuture<DeleteClusterResponse>
 deleteRedshiftClusterAsync(String clusterId) {
        DeleteClusterRequest deleteClusterRequest = DeleteClusterRequest.builder()
            .clusterIdentifier(clusterId)
            .skipFinalClusterSnapshot(true)
            .build();

        return getAsyncClient().deleteCluster(deleteClusterRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    // Handle exceptions
                    if (exception.getCause() instanceof RedshiftException) {
                        logger.info("Error: {}", exception.getMessage());
                    } else {
                        logger.info("Unexpected error: {}", exception.getMessage());
                    }
                } else {
                    // Handle successful response
                    logger.info("The status is {}",
 response.cluster().clusterStatus());
                }
            });
    }
}
```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella Documentazione di riferimento delle API AWS SDK for Java 2.x .
  - CreateCluster
  - DescribeClusters
  - DescribeStatement
  - ExecuteStatement
  - GetStatementResult
  - ListDatabasesPaginator
  - ModifyCluster

Azioni

`CreateCluster`

Il seguente esempio di codice mostra come usare. `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 .

```
   /**
    * Creates a new Amazon Redshift cluster asynchronously.
    * @param clusterId     the unique identifier for the cluster
    * @param username      the username for the administrative user
    * @param userPassword  the password for the administrative user
    * @return a CompletableFuture that represents the asynchronous operation of
creating the cluster
    * @throws RuntimeException if the cluster creation fails
    */
   public CompletableFuture<CreateClusterResponse> createClusterAsync(String
clusterId, String username, String userPassword) {
       CreateClusterRequest clusterRequest = CreateClusterRequest.builder()
           .clusterIdentifier(clusterId)
           .masterUsername(username)
           .masterUserPassword(userPassword)
           .nodeType("ra3.4xlarge")
           .publiclyAccessible(true)
           .numberOfNodes(2)
           .build();

       return getAsyncClient().createCluster(clusterRequest)
           .whenComplete((response, exception) -> {
               if (response != null) {
                   logger.info("Created cluster ");
               } else {
                   throw new RuntimeException("Failed to create cluster: " +
exception.getMessage(), exception);
```

```
                }
            });
    }
```

- Per i dettagli sull'API, consulta la CreateClustersezione 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.

```
    /**
     * Deletes a Redshift cluster asynchronously.
     *
     * @param clusterId the identifier of the Redshift cluster to be deleted
     * @return a {@link CompletableFuture} that represents the asynchronous
operation of deleting the Redshift cluster
     */
    public CompletableFuture<DeleteClusterResponse>
deleteRedshiftClusterAsync(String clusterId) {
        DeleteClusterRequest deleteClusterRequest = DeleteClusterRequest.builder()
            .clusterIdentifier(clusterId)
            .skipFinalClusterSnapshot(true)
            .build();

        return getAsyncClient().deleteCluster(deleteClusterRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    // Handle exceptions
                    if (exception.getCause() instanceof RedshiftException) {
                        logger.info("Error: {}", exception.getMessage());
                    } else {
```

```
                    logger.info("Unexpected error: {}", exception.getMessage());
                }
            } else {
                // Handle successful response
                logger.info("The status is {}",
response.cluster().clusterStatus());
            }
        });
   }
```

- Per i dettagli sull'API, consulta la DeleteClustersezione 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.

```
   /**
    * Waits asynchronously for the specified cluster to become available.
    * @param clusterId the identifier of the cluster to wait for
    * @return a {@link CompletableFuture} that completes when the cluster is ready
    */
   public CompletableFuture<Void> waitForClusterReadyAsync(String clusterId) {
       DescribeClustersRequest clustersRequest = DescribeClustersRequest.builder()
           .clusterIdentifier(clusterId)
           .build();

       logger.info("Waiting for cluster to become available. This may take a few
minutes.");
       long startTime = System.currentTimeMillis();

       // Recursive method to poll the cluster status.
```

```
        return checkClusterStatusAsync(clustersRequest, startTime);
    }

    private CompletableFuture<Void> checkClusterStatusAsync(DescribeClustersRequest
clustersRequest, long startTime) {
        return getAsyncClient().describeClusters(clustersRequest)
            .thenCompose(clusterResponse -> {
                List<Cluster> clusterList = clusterResponse.clusters();
                boolean clusterReady = false;
                for (Cluster cluster : clusterList) {
                    if ("available".equals(cluster.clusterStatus())) {
                        clusterReady = true;
                        break;
                    }
                }

                if (clusterReady) {
                    logger.info(String.format("Cluster is available!"));
                    return CompletableFuture.completedFuture(null);
                } else {
                    long elapsedTimeMillis = System.currentTimeMillis() - startTime;
                    long elapsedSeconds = elapsedTimeMillis / 1000;
                    long minutes = elapsedSeconds / 60;
                    long seconds = elapsedSeconds % 60;
                    System.out.printf("\rElapsed Time: %02d:%02d - Waiting for
cluster...", minutes, seconds);
                    System.out.flush();

                    // Wait 1 second before the next status check
                    return CompletableFuture.runAsync(() -> {
                        try {
                            TimeUnit.SECONDS.sleep(1);
                        } catch (InterruptedException e) {
                            throw new RuntimeException("Error during sleep: " +
e.getMessage(), e);
                        }
                    }).thenCompose(ignored ->
checkClusterStatusAsync(clustersRequest, startTime));
                }
            }).exceptionally(exception -> {
                throw new RuntimeException("Failed to get cluster status: " +
exception.getMessage(), exception);
            });
    }
```

- Per i dettagli sull'API, DescribeClustersconsulta 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.

```
   /**
    * Checks the status of an SQL statement asynchronously and handles the
completion of the statement.
    *
    * @param sqlId the ID of the SQL statement to check
    * @return a {@link CompletableFuture} that completes when the SQL statement's
status is either "FINISHED" or "FAILED"
    */
   public CompletableFuture<Void> checkStatementAsync(String sqlId) {
       DescribeStatementRequest statementRequest =
DescribeStatementRequest.builder()
           .id(sqlId)
           .build();

       return getAsyncDataClient().describeStatement(statementRequest)
           .thenCompose(response -> {
               String status = response.statusAsString();
               logger.info("... Status: {} ", status);

               if ("FAILED".equals(status)) {
                   throw new RuntimeException("The Query Failed. Ending program");
               } else if ("FINISHED".equals(status)) {
                   return CompletableFuture.completedFuture(null);
               } else {
                   // Sleep for 1 second and recheck status
                   return CompletableFuture.runAsync(() -> {
```

```
                        try {
                            TimeUnit.SECONDS.sleep(1);
                        } catch (InterruptedException e) {
                            throw new RuntimeException("Error during sleep: " +
e.getMessage(), e);
                        }
                    }).thenCompose(ignore -> checkStatementAsync(sqlId)); //
Recursively call until status is FINISHED or FAILED
                }
            }).whenComplete((result, exception) -> {
                if (exception != null) {
                    // Handle exceptions
                    logger.info("Error: {} ", exception.getMessage());
                } else {
                    logger.info("The statement is finished!");
                }
            });
    }
```

- Per i dettagli sull'API, consulta la DescribeStatementsezione AWS SDK for Java 2.x API Reference.

`ExecuteStatement`

Il seguente esempio di codice mostra come utilizzare`ExecuteStatement`.

SDK per Java 2.x

> **Note**
>
> C'è 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 un'istruzione SQL per creare una tabella di database.

```
    /**
     * Creates an asynchronous task to execute a SQL statement for creating a new
table.
     *
     * @param clusterId    the identifier of the Amazon Redshift cluster
```

```
    * @param databaseName the name of the database to create the table in
    * @param userName     the username to use for the database connection
    * @return a {@link CompletableFuture} that completes with the result of the SQL
statement execution
    * @throws RuntimeException if there is an error creating the table
    */
   public CompletableFuture<ExecuteStatementResponse> createTableAsync(String
clusterId, String databaseName, String userName) {
       ExecuteStatementRequest createTableRequest =
ExecuteStatementRequest.builder()
           .clusterIdentifier(clusterId)
           .dbUser(userName)
           .database(databaseName)
           .sql("CREATE TABLE Movies (" +
               "id INT PRIMARY KEY, " +
               "title VARCHAR(100), " +
               "year INT)")
           .build();

       return getAsyncDataClient().executeStatement(createTableRequest)
           .whenComplete((response, exception) -> {
               if (exception != null) {
                   throw new RuntimeException("Error creating table: " +
exception.getMessage(), exception);
               } else {
                   logger.info("Table created: Movies");
               }
           });
   }
```

Esegue un'istruzione SQL per inserire dati in una tabella di database.

```
   /**
    * Asynchronously pops a table from a JSON file.
    *
    * @param clusterId   the ID of the cluster
    * @param databaseName the name of the database
    * @param userName    the username
    * @param fileName    the name of the JSON file
    * @param number      the number of records to process
    * @return a CompletableFuture that completes with the number of records added
to the Movies table
```

```
    */
   public CompletableFuture<Integer> popTableAsync(String clusterId, String
databaseName, String userName, String fileName, int number) {
       return CompletableFuture.supplyAsync(() -> {
               try {
                   JsonParser parser = new JsonFactory().createParser(new
File(fileName));
                   JsonNode rootNode = new ObjectMapper().readTree(parser);
                   Iterator<JsonNode> iter = rootNode.iterator();
                   return iter;
               } catch (IOException e) {
                   throw new RuntimeException("Failed to read or parse JSON file: "
+ e.getMessage(), e);
               }
           }).thenCompose(iter -> processNodesAsync(clusterId, databaseName,
userName, iter, number))
           .whenComplete((result, exception) -> {
               if (exception != null) {
                   logger.info("Error {} ", exception.getMessage());
               } else {
                   logger.info("{} records were added to the Movies table." ,
result);
               }
           });
   }

   private CompletableFuture<Integer> processNodesAsync(String clusterId, String
databaseName, String userName, Iterator<JsonNode> iter, int number) {
       return CompletableFuture.supplyAsync(() -> {
           int t = 0;
           try {
               while (iter.hasNext()) {
                   if (t == number)
                       break;
                   JsonNode currentNode = 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);";
                   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);

                ExecuteStatementRequest insertStatementRequest =
ExecuteStatementRequest.builder()
                        .clusterIdentifier(clusterId)
                        .sql(sqlStatement)
                        .database(databaseName)
                        .dbUser(userName)
                        .parameters(parameterList)
                        .build();

                getAsyncDataClient().executeStatement(insertStatementRequest);
                logger.info("Inserted: " + title + " (" + year + ")");
                t++;
            }
        } catch (RedshiftDataException e) {
            throw new RuntimeException("Error inserting data: " +
e.getMessage(), e);
        }
        return t;
    });
  }
```

Esegue un'istruzione SQL per interrogare una tabella di database.

```
  /**
   * Asynchronously queries movies by a given year from a Redshift database.
   *
```

```
     * @param database    the name of the database to query
     * @param dbUser      the user to connect to the database with
     * @param year        the year to filter the movies by
     * @param clusterId   the identifier of the Redshift cluster to connect to
     * @return a {@link CompletableFuture} containing the response ID of the
executed SQL statement
     */
    public CompletableFuture<String> queryMoviesByYearAsync(String database,
                                                            String dbUser,
                                                            int year,
                                                            String clusterId)
{

        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();

        return CompletableFuture.supplyAsync(() -> {
            try {
                ExecuteStatementResponse response =
getAsyncDataClient().executeStatement(statementRequest).join(); // Use join() to
wait for the result
                return response.id();
            } catch (RedshiftDataException e) {
                throw new RuntimeException("Error executing statement: " +
e.getMessage(), e);
            }
        }).exceptionally(exception -> {
            logger.info("Error: {}", exception.getMessage());
            return "";
        });
    }
```

- Per i dettagli sull'API, vedere ExecuteStatementin 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.

```
   /**
    * Asynchronously retrieves the results of a statement execution.
    *
    * @param statementId the ID of the statement for which to retrieve the results
    * @return a {@link CompletableFuture} that completes when the statement result
has been processed
    */
   public CompletableFuture<Void> getResultsAsync(String statementId) {
       GetStatementResultRequest resultRequest =
GetStatementResultRequest.builder()
           .id(statementId)
           .build();

       return getAsyncDataClient().getStatementResult(resultRequest)
           .handle((response, exception) -> {
               if (exception != null) {
                   logger.info("Error getting statement result {} ",
exception.getMessage());
                   throw new RuntimeException("Error getting statement result: " +
exception.getMessage(), exception);
               }

               // Extract and print the field values using streams if the response
is valid.
               response.records().stream()
                   .flatMap(List::stream)
```

```
                    .map(Field::stringValue)
                    .filter(value -> value != null)
                    .forEach(value -> System.out.println("The Movie title field is "
+ value));

                return response;
            }).thenAccept(response -> {
                // Optionally add more logic here if needed after handling the
response
            });
    }
```

- Per i dettagli sull'API, consulta la GetStatementResultsezione AWS SDK for Java 2.x API Reference.

`ListDatabases`

Il seguente esempio di codice mostra come utilizzare`ListDatabases`.

SDK per Java 2.x

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
    /**
     * Lists all databases asynchronously for the specified cluster, database user,
and database.
     * @param clusterId the identifier of the cluster to list databases for
     * @param dbUser the database user to use for the list databases request
     * @param database the database to list databases for
     * @return a {@link CompletableFuture} that completes when the database listing
is complete, or throws a {@link RuntimeException} if there was an error
     */
    public CompletableFuture<Void> listAllDatabasesAsync(String clusterId, String
dbUser, String database) {
        ListDatabasesRequest databasesRequest = ListDatabasesRequest.builder()
            .clusterIdentifier(clusterId)
```

```
            .dbUser(dbUser)
            .database(database)
            .build();

        // Asynchronous paginator for listing databases.
        ListDatabasesPublisher databasesPaginator =
getAsyncDataClient().listDatabasesPaginator(databasesRequest);
        CompletableFuture<Void> future = databasesPaginator.subscribe(response -> {
            response.databases().forEach(db -> {
                logger.info("The database name is {} ", db);
            });
        });

        // Return the future for asynchronous handling.
        return future.exceptionally(exception -> {
            throw new RuntimeException("Failed to list databases: " +
exception.getMessage(), exception);
        });
    }
```

- Per i dettagli sull'API, consulta la ListDatabasessezione 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.

```
    /**
     * Modifies an Amazon Redshift cluster asynchronously.
     *
     * @param clusterId the identifier of the cluster to be modified
```

```
     * @return a {@link CompletableFuture} that completes when the cluster
modification is complete
     */
    public CompletableFuture<ModifyClusterResponse> modifyClusterAsync(String
clusterId) {
        ModifyClusterRequest modifyClusterRequest = ModifyClusterRequest.builder()
            .clusterIdentifier(clusterId)
            .preferredMaintenanceWindow("wed:07:30-wed:08:00")
            .build();

        return getAsyncClient().modifyCluster(modifyClusterRequest)
            .whenComplete((clusterResponse, exception) -> {
                if (exception != null) {
                    if (exception.getCause() instanceof RedshiftException) {
                        logger.info("Error: {} ", exception.getMessage());
                    } else {
                        logger.info("Unexpected error: {} ",
exception.getMessage());
                    }
                } else {
                    logger.info("The modified cluster was successfully modified and
has "
                        + clusterResponse.cluster().preferredMaintenanceWindow() + "
as the maintenance window");
                }
            });
    }
```

- Per i dettagli sull'API, ModifyClusterconsulta AWS SDK for Java 2.x API Reference.

Scenari

Creazione di un'applicazione Web per tracciare i dati Amazon Redshift

Il seguente esempio di codice mostra come creare un'applicazione Web che traccia e genera report sugli elementi di lavoro utilizzando un database 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

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.

Gli scenari sono esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio o combinate con altri Servizi AWS.

Ogni esempio include un collegamento al codice sorgente completo, 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.*;
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.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * 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: <bucketName> <sourceKey> <targetKey>

            Where:
                bucketName - The name of the S3 bucket where the images are stored.
                sourceKey  - The S3 key (file name) for the source image.
                targetKey  - The S3 key (file name) for the target image.
           """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }
```

```
        String bucketName = args[0];
        String sourceKey = args[1];
        String targetKey = args[2];

        Region region = Region.US_WEST_2;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();
        compareTwoFaces(rekClient, bucketName, sourceKey, targetKey);
    }

   /**
    * Compares two faces from images stored in an Amazon S3 bucket using AWS
Rekognition.
    *
    * <p>This method takes two image keys from an S3 bucket and compares the faces
within them.
    * It prints out the confidence level of matched faces and reports the number of
unmatched faces.</p>
    *
    * @param rekClient   The {@link RekognitionClient} used to call AWS
Rekognition.
    * @param bucketName  The name of the S3 bucket containing the images.
    * @param sourceKey   The object key (file path) for the source image in the S3
bucket.
    * @param targetKey   The object key (file path) for the target image in the S3
bucket.
    * @throws RuntimeException If the Rekognition service returns an error.
    */
   public static void compareTwoFaces(RekognitionClient rekClient, String
bucketName, String sourceKey, String targetKey) {
        try {
            Float similarityThreshold = 70F;
            S3Object s3ObjectSource = S3Object.builder()
                    .bucket(bucketName)
                    .name(sourceKey)
                    .build();

            Image sourceImage = Image.builder()
                    .s3Object(s3ObjectSource)
                    .build();

            S3Object s3ObjectTarget = S3Object.builder()
```

```
                    .bucket(bucketName)
                    .name(targetKey)
                    .build();

            Image targetImage = Image.builder()
                    .s3Object(s3ObjectTarget)
                    .build();

            CompareFacesRequest facesRequest = CompareFacesRequest.builder()
                    .sourceImage(sourceImage)
                    .targetImage(targetImage)
                    .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> unmatchedFaces = compareFacesResult.unmatchedFaces();
            System.out.println("There were " + unmatchedFaces.size() + " face(s)
 that did not match.");

        } catch (RekognitionException e) {
            System.err.println("Error comparing faces: " +
 e.awsErrorDetails().errorMessage());
            throw new RuntimeException(e);
        }
    }
}
```

- Per i dettagli sull'API, consulta la CompareFacessezione 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_WEST_2;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        System.out.println("Creating collection: " + collectionId);
        createMyCollection(rekClient, collectionId);
        rekClient.close();
    }

    /**
     * Creates a new Amazon Rekognition collection.
     *
     * @param rekClient    the Amazon Rekognition client used to interact with the
 Rekognition service
     * @param collectionId the unique identifier for the collection to be created
     */
    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 CreateCollectionsezione 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();
    }

    /**
     * Deletes an Amazon Rekognition collection.
     *
     * @param rekClient      An instance of the {@link RekognitionClient} class,
 which is used to interact with the Amazon Rekognition service.
     * @param collectionId   The ID of the collection to be deleted.
     */
    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 DeleteCollectionsezione 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 != 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("Deleting collection: " + collectionId);
        deleteFacesCollection(rekClient, collectionId, faceId);
        rekClient.close();
    }

    /**
     * Deletes a face from the specified Amazon Rekognition collection.
     *
     * @param rekClient     an instance of the Amazon Rekognition client
     * @param collectionId  the ID of the collection from which the face should be
 deleted
     * @param faceId        the ID of the face to be deleted
     * @throws RekognitionException if an error occurs while deleting the face
     */
    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 DeleteFacessezione 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();
    }

    /**
     * Describes an Amazon Rekognition collection.
     *
     * @param rekClient         The Amazon Rekognition client used to make the
 request.
     * @param collectionName    The name of the collection to describe.
     *
     * @throws RekognitionException If an error occurs while describing the
 collection.
     */
    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 DescribeCollectionsezione 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.*;

import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * 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:   <bucketName> <sourceImage>

            Where:
                bucketName = The name of the Amazon S3 bucket where the source image
 is stored.
```

```
                sourceImage - The name of the source image file in the Amazon S3
bucket. (for example, pic1.png).\s
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String sourceImage = args[1];
        Region region = Region.US_WEST_2;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        detectFacesinImage(rekClient, bucketName, sourceImage);
        rekClient.close();
    }

    /**
     * Detects faces in an image stored in an Amazon S3 bucket using the Amazon
Rekognition service.
     *
     * @param rekClient    The Amazon Rekognition client used to interact with the
Rekognition service.
     * @param bucketName   The name of the Amazon S3 bucket where the source image
is stored.
     * @param sourceImage  The name of the source image file in the Amazon S3
bucket.
     */
    public static void detectFacesinImage(RekognitionClient rekClient, String
bucketName, String sourceImage) {
        try {
            S3Object s3ObjectTarget = S3Object.builder()
                .bucket(bucketName)
                .name(sourceImage)
                .build();

            Image targetImage = Image.builder()
                .s3Object(s3ObjectTarget)
                .build();

            DetectFacesRequest facesRequest = DetectFacesRequest.builder()
```

```
                .attributes(Attribute.ALL)
                .image(targetImage)
                .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 e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Per i dettagli sull'API, consulta la DetectFacessezione 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.*;

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: <bucketName> <sourceImage>

            Where:
                bucketName - The name of the Amazon S3 bucket where the image is
 stored
                sourceImage - The name of the image file (for example, pic1.png).\s
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0] ;
        String sourceImage = args[1] ;
        Region region = Region.US_WEST_2;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        detectImageLabels(rekClient, bucketName, sourceImage);
        rekClient.close();
    }
```

```
    /**
     * Detects the labels in an image stored in an Amazon S3 bucket using the Amazon
 Rekognition service.
     *
     * @param rekClient     the Amazon Rekognition client used to make the detection
 request
     * @param bucketName    the name of the Amazon S3 bucket where the image is
 stored
     * @param sourceImage   the name of the image file to be analyzed
     */
    public static void detectImageLabels(RekognitionClient rekClient, String
 bucketName, String sourceImage) {
        try {
            S3Object s3ObjectTarget = S3Object.builder()
                    .bucket(bucketName)
                    .name(sourceImage)
                    .build();

            Image souImage = Image.builder()
                    .s3Object(s3ObjectTarget)
                    .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 e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Per i dettagli sull'API, consulta la DetectLabelssezione 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.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.*;

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:  <bucketName>  <sourceImage>

            Where:
                bucketName - The name of the S3 bucket where the images are stored.
                sourceImage - The name of the image (for example, pic1.png).\s
```

```
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String sourceImage = args[1];
        Region region = Region.US_WEST_2;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        detectModLabels(rekClient, bucketName, sourceImage);
        rekClient.close();
    }

    /**
     * Detects moderation labels in an image stored in an Amazon S3 bucket.
     *
     * @param rekClient      the Amazon Rekognition client to use for the detection
     * @param bucketName     the name of the Amazon S3 bucket where the image is
stored
     * @param sourceImage    the name of the image file to be analyzed
     *
     * @throws RekognitionException if there is an error during the image detection
process
     */
    public static void detectModLabels(RekognitionClient rekClient, String
bucketName, String sourceImage) {
        try {
            S3Object s3ObjectTarget = S3Object.builder()
                    .bucket(bucketName)
                    .name(sourceImage)
                    .build();

            Image targetImage = Image.builder()
                    .s3Object(s3ObjectTarget)
                    .build();

            DetectModerationLabelsRequest moderationLabelsRequest =
DetectModerationLabelsRequest.builder()
                    .image(targetImage)
```

```
                    .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 e) {
            e.printStackTrace();
            System.exit(1);
        }
    }
}
```

- Per i dettagli sull'API, consulta la DetectModerationLabelssezione 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.*;

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 = "\n" +
            "Usage:   <bucketName> <sourceImage>\n" +
            "\n" +
            "Where:\n" +
            "   bucketName - The name of the S3 bucket where the image is stored\n"
 +
            "   sourceImage - The path to the image that contains text (for example,
 pic1.png). \n";

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String sourceImage = args[1];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        detectTextLabels(rekClient, bucketName, sourceImage);
        rekClient.close();
    }

    /**
```

```
    * Detects text labels in an image stored in an S3 bucket using Amazon
Rekognition.
    *
    * @param rekClient    an instance of the Amazon Rekognition client
    * @param bucketName   the name of the S3 bucket where the image is stored
    * @param sourceImage  the name of the image file in the S3 bucket
    * @throws RekognitionException if an error occurs while calling the Amazon
Rekognition API
    */
   public static void detectTextLabels(RekognitionClient rekClient, String
bucketName, String sourceImage) {
       try {
           S3Object s3ObjectTarget = S3Object.builder()
                   .bucket(bucketName)
                   .name(sourceImage)
                   .build();

           Image souImage = Image.builder()
                   .s3Object(s3ObjectTarget)
                   .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 e) {
           System.out.println(e.getMessage());
           System.exit(1);
       }
   }
}
```

- Per i dettagli sull'API, consulta la DetectTextsezione 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.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.*;
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> <bucketName>

            Where:
                collectionName - The name of the collection.
                sourceImage - The name of the image (for example, pic1.png).
                bucketName - The name of the S3 bucket.
            """;
```

```
        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String collectionId = args[0];
        String sourceImage = args[1];
        String bucketName = args[2];;
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        addToCollection(rekClient, collectionId, bucketName, sourceImage);
        rekClient.close();
    }

    /**
     * Adds a face from an image to an Amazon Rekognition collection.
     *
     * @param rekClient     the Amazon Rekognition client
     * @param collectionId  the ID of the collection to add the face to
     * @param bucketName    the name of the Amazon S3 bucket containing the image
     * @param sourceImage   the name of the image file to add to the collection
     * @throws RekognitionException if there is an error while interacting with the
Amazon Rekognition service
     */
    public static void addToCollection(RekognitionClient rekClient, String
collectionId, String bucketName, String sourceImage) {
        try {
            S3Object s3ObjectTarget = S3Object.builder()
                    .bucket(bucketName)
                    .name(sourceImage)
                    .build();

            Image targetImage = Image.builder()
                    .s3Object(s3ObjectTarget)
                    .build();

            IndexFacesRequest facesRequest = IndexFacesRequest.builder()
                    .collectionId(collectionId)
                    .image(targetImage)
                    .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 e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Per i dettagli sull'API, consulta la IndexFacessezione 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 ListCollectionssezione 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 ListFacessezione 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.*;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * 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:   <bucketName> <sourceImage>

                Where:
                   bucketName - The name of the S3 bucket where the images are
 stored.
                   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 bucketName = args[0];;
        String sourceImage = args[1];
        Region region = Region.US_WEST_2;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        System.out.println("Locating celebrities in " + sourceImage);
        recognizeAllCelebrities(rekClient, bucketName, sourceImage);
        rekClient.close();
    }

    /**
     * Recognizes all celebrities in an image stored in an Amazon S3 bucket.
     *
     * @param rekClient    the Amazon Rekognition client used to perform the
 celebrity recognition operation
     * @param bucketName   the name of the Amazon S3 bucket where the source image
 is stored
     * @param sourceImage  the name of the source image file stored in the Amazon S3
 bucket
     */
    public static void recognizeAllCelebrities(RekognitionClient rekClient, String
 bucketName, String sourceImage) {
```

```
        try {
            S3Object s3ObjectTarget = S3Object.builder()
                .bucket(bucketName)
                .name(sourceImage)
                .build();

            Image souImage = Image.builder()
                    .s3Object(s3ObjectTarget)
                    .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 e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Per i dettagli sull'API, consulta la RecognizeCelebritiessezione 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_WEST_2;
        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 SearchFacessezione 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_WEST_2;
        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 SearchFacesByImagesezione AWS SDK for Java 2.x API Reference.

Scenari

Creazione di un'applicazione serverless per gestire foto

Nell'esempio di codice seguente viene illustrato come creare un'applicazione serverless che consente agli utenti di gestire le foto mediante 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

Rilevamento dei DPI nelle immagini

Il seguente esempio di codice mostra come creare un'app che utilizza Amazon Rekognition per rilevare i dispositivi di protezione individuale (DPI) nelle immagini.

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

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 altro 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 vidOb = Video.builder()
                    .s3Object(s3Obj)
                    .build();

            StartCelebrityRecognitionRequest recognitionRequest =
StartCelebrityRecognitionRequest.builder()
                    .jobTag("Celebrities")
                    .notificationChannel(channel)
                    .video(vidOb)
                    .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 vidOb = Video.builder()
                    .s3Object(s3Obj)
                    .build();

            StartLabelDetectionRequest labelDetectionRequest =
StartLabelDetectionRequest.builder()
                    .jobTag("DetectingLabels")
                    .notificationChannel(channel)
                    .video(vidOb)
                    .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(RekognitionClient rekClient,
            NotificationChannel channel,
            String bucket,
            String video) {
        try {
            S3Object s3Obj = S3Object.builder()
                    .bucket(bucket)
                    .name(video)
                    .build();

            Video vidOb = Video.builder()
                    .s3Object(s3Obj)
                    .build();

            StartLabelDetectionRequest labelDetectionRequest =
StartLabelDetectionRequest.builder()
                    .jobTag("DetectingLabels")
                    .notificationChannel(channel)
                    .video(vidOb)
                    .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 vidOb = Video.builder()
                    .s3Object(s3Obj)
                    .build();

            StartContentModerationRequest modDetectionRequest =
StartContentModerationRequest.builder()
                    .jobTag("Moderation")
                    .notificationChannel(channel)
                    .video(vidOb)
                    .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 vidOb = 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(RekognitionClient rekClient,
            NotificationChannel channel,
            String bucket,
            String video) {
        try {
            S3Object s3Obj = S3Object.builder()
                    .bucket(bucket)
                    .name(video)
                    .build();

            Video vidOb = Video.builder()
                    .s3Object(s3Obj)
                    .build();

            StartTextDetectionRequest labelDetectionRequest =
StartTextDetectionRequest.builder()
                    .jobTag("DetectingLabels")
                    .notificationChannel(channel)
                    .video(vidOb)
                    .build();

            StartTextDetectionResponse labelDetectionResponse =
rekClient.startTextDetection(labelDetectionRequest);
            startJobId = labelDetectionResponse.jobId();

        } catch (RekognitionException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
```

```
    }

    public static void getTextResults(RekognitionClient rekClient) {
        try {
            String paginationToken = null;
            GetTextDetectionResponse textDetectionResponse = null;
            boolean finished = false;
            String status;
            int yy = 0;

            do {
                if (textDetectionResponse != null)
                    paginationToken = textDetectionResponse.nextToken();

                GetTextDetectionRequest recognitionRequest =
GetTextDetectionRequest.builder()
                        .jobId(startJobId)
                        .nextToken(paginationToken)
                        .maxResults(10)
                        .build();

                // Wait until the job succeeds.
                while (!finished) {
                    textDetectionResponse =
rekClient.getTextDetection(recognitionRequest);
                    status = textDetectionResponse.jobStatusAsString();

                    if (status.compareTo("SUCCEEDED") == 0)
                        finished = true;
                    else {
                        System.out.println(yy + " status is: " + status);
                        Thread.sleep(1000);
                    }
                    yy++;
                }

                finished = false;

                // Proceed when the job is done - otherwise VideoMetadata is null.
                VideoMetadata videoMetaData = textDetectionResponse.videoMetadata();
                System.out.println("Format: " + videoMetaData.format());
                System.out.println("Codec: " + videoMetaData.codec());
                System.out.println("Duration: " + videoMetaData.durationMillis());
                System.out.println("FrameRate: " + videoMetaData.frameRate());
```

```
                System.out.println("Job");

                List<TextDetectionResult> labels =
 textDetectionResponse.textDetections();
                for (TextDetectionResult detectedText : labels) {
                    System.out.println("Confidence: " +
 detectedText.textDetection().confidence().toString());
                    System.out.println("Id : " + detectedText.textDetection().id());
                    System.out.println("Parent Id: " +
 detectedText.textDetection().parentId());
                    System.out.println("Type: " +
 detectedText.textDetection().type());
                    System.out.println("Text: " +
 detectedText.textDetection().detectedText());
                    System.out.println();
                }

            } while (textDetectionResponse != null &&
 textDetectionResponse.nextToken() != null);

        } catch (RekognitionException | InterruptedException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

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(RekognitionClient rekClient,
            NotificationChannel channel,
            String bucket,
            String video) {
        try {
            S3Object s3Obj = S3Object.builder()
                    .bucket(bucket)
                    .name(video)
                    .build();

            Video vidOb = Video.builder()
                    .s3Object(s3Obj)
                    .build();

            StartPersonTrackingRequest personTrackingRequest =
StartPersonTrackingRequest.builder()
                    .jobTag("DetectingLabels")
                    .video(vidOb)
                    .notificationChannel(channel)
                    .build();

            StartPersonTrackingResponse labelDetectionResponse =
rekClient.startPersonTracking(personTrackingRequest);
            startJobId = labelDetectionResponse.jobId();

        } catch (RekognitionException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void getPersonDetectionResults(RekognitionClient rekClient) {
        try {
            String paginationToken = null;
```

```
            GetPersonTrackingResponse personTrackingResult = null;
            boolean finished = false;
            String status;
            int yy = 0;

            do {
                if (personTrackingResult != null)
                    paginationToken = personTrackingResult.nextToken();

                GetPersonTrackingRequest recognitionRequest =
GetPersonTrackingRequest.builder()
                        .jobId(startJobId)
                        .nextToken(paginationToken)
                        .maxResults(10)
                        .build();

                // Wait until the job succeeds
                while (!finished) {

                    personTrackingResult =
rekClient.getPersonTracking(recognitionRequest);
                    status = personTrackingResult.jobStatusAsString();

                    if (status.compareTo("SUCCEEDED") == 0)
                        finished = true;
                    else {
                        System.out.println(yy + " status is: " + status);
                        Thread.sleep(1000);
                    }
                    yy++;
                }

                finished = false;

                // Proceed when the job is done - otherwise VideoMetadata is null.
                VideoMetadata videoMetaData = personTrackingResult.videoMetadata();

                System.out.println("Format: " + videoMetaData.format());
                System.out.println("Codec: " + videoMetaData.codec());
                System.out.println("Duration: " + videoMetaData.durationMillis());
                System.out.println("FrameRate: " + videoMetaData.frameRate());
                System.out.println("Job");
```

```
                List<PersonDetection> detectedPersons =
 personTrackingResult.persons();
                for (PersonDetection detectedPerson : detectedPersons) {
                    long seconds = detectedPerson.timestamp() / 1000;
                    System.out.print("Sec: " + seconds + " ");
                    System.out.println("Person Identifier: " +
 detectedPerson.person().index());
                    System.out.println();
                }

            } while (personTrackingResult != null &&
 personTrackingResult.nextToken() != null);

        } catch (RekognitionException | InterruptedException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- 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

Rilevamento di oggetti nelle immagini

Il seguente esempio di codice mostra come creare un'app che utilizza Amazon Rekognition per rilevare oggetti per categoria nelle immagini.

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

Rilevamento di persone e oggetti in un video

Il seguente esempio di codice mostra come rilevare persone e oggetti in un video con Amazon Rekognition.

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

- Amazon SNS
- Amazon SQS

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 nozioni di base sono esempi di codice che mostrano come eseguire le operazioni essenziali all'interno di un servizio.

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.

Ogni esempio include un collegamento al codice sorgente completo, in cui 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 ListPricessezione AWS SDK for Java 2.x API Reference.

Argomenti

- Nozioni di base
- Azioni

Nozioni di base

Informazioni di base

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 xxxxxxx not found in xxxxxxx 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

Azioni

`CheckDomainAvailability`

Il seguente esempio di codice mostra come usare`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 CheckDomainAvailabilitysezione 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 CheckDomainTransferabilitysezione 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 GetDomainDetailsezione 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 GetDomainSuggestionssezione 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 GetOperationDetailsezione 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 ListDomainssezione 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 ListOperationssezione 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 ListPricessezione 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 RegisterDomainsezione 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 ViewBillingsezione AWS SDK for Java 2.x API Reference.

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 nozioni di base sono esempi di codice che mostrano come eseguire le operazioni essenziali all'interno di un servizio.

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.

Gli scenari sono esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio o combinate con altri Servizi AWS.

Ogni esempio include un collegamento al codice sorgente completo, 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.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * 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);
    }

    /**
     * Lists all the S3 buckets associated with the provided AWS S3 client.
     *
```

```
     * @param s3 the S3Client instance used to interact with the AWS S3 service
     */
    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, ListBucketsconsulta AWS SDK for Java 2.x API Reference.

Argomenti

- Nozioni di base
- Azioni
- Scenari
- Esempi serverless

Nozioni di base

Informazioni di base

L'esempio di codice seguente mostra come:

- Crea un bucket e carica 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.
- Elimina il bucket e tutti gli oggetti in esso contenuti.

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.

Un esempio di scenario.

```
import java.io.IOException;
import java.util.Scanner;
import java.util.UUID;
import java.util.concurrent.CompletableFuture;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.s3.model.PutObjectResponse;
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
 *
 * 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. Copy the object to another Amazon S3 bucket using multi copy.
 * 8. Deletes the object from the Amazon S3 bucket.
 * 9. Deletes the Amazon S3 bucket.
 */

public class S3Scenario {
```

```
   public static Scanner scanner = new Scanner(System.in);
   static S3Actions s3Actions = new S3Actions();
   public static final String DASHES = new String(new char[80]).replace("\0", "-");
   private static final Logger logger = LoggerFactory.getLogger(S3Scenario.class);
   public static void main(String[] args) throws IOException {
       final String usage = """
           Usage:
              <bucketName> <key> <objectPath> <savePath> <toBucket>

           Where:
               bucketName - The name of the  S3 bucket.
               key - The unique identifier for the object stored in the S3 bucket.
               objectPath - The full file path of the object within the S3 bucket
(e.g., "documents/reports/annual_report.pdf").
               savePath - The local file path where the object will be downloaded
and saved (e.g., "C:/Users/username/Downloads/annual_report.pdf").
               toBucket - The name of the S3 bucket to which the object will be
copied.
           """;

       if (args.length != 5) {
           logger.info(usage);
           return;
       }

       String bucketName = args[0];
       String key = args[1];
       String objectPath = args[2];
       String savePath = args[3];
       String toBucket = args[4];

       logger.info(DASHES);
       logger.info("Welcome to the Amazon Simple Storage Service (S3) example
scenario.");
       logger.info("""
           Amazon S3 is a highly scalable and durable object storage
           service provided by Amazon Web Services (AWS). It is designed to store
and retrieve
           any amount of data, from anywhere on the web, at any time.

           The `S3AsyncClient` interface in the AWS SDK for Java 2.x provides a set
of methods to
           programmatically interact with the Amazon S3 (Simple Storage Service)
service. This allows
```

```
            developers to automate the management and manipulation of S3 buckets and
objects as
            part of their application deployment pipelines. With S3, teams can focus
on building
            and deploying their applications without having to worry about the
underlying storage
            infrastructure required to host and manage large amounts of data.

            This scenario walks you through how to perform key operations for this
service.
            Let's get started...
            """);
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        try {
            // Run the methods that belong to this scenario.
            runScenario(bucketName, key, objectPath, savePath, toBucket);

        } catch (Throwable rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof S3Exception kmsEx) {
                logger.info("KMS error occurred: Error message: {}, Error code {}",
kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
        }
    }

    private static void runScenario(String bucketName, String key, String
objectPath, String savePath, String toBucket) throws Throwable {
        logger.info(DASHES);
        logger.info("1. Create an Amazon S3 bucket.");
        try {
            CompletableFuture<Void> future =
s3Actions.createBucketAsync(bucketName);
            future.join();
            waitForInputToContinue(scanner);

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof S3Exception s3Ex) {
```

```
                logger.info("S3 error occurred: Error message: {}, Error code {}",
s3Ex.getMessage(), s3Ex.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            throw cause;

        }
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("2. Upload a local file to the Amazon S3 bucket.");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<PutObjectResponse> future =
s3Actions.uploadLocalFileAsync(bucketName, key, objectPath);
            future.join();
            logger.info("File uploaded successfully to {}/{}", bucketName, key);

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof S3Exception s3Ex) {
                logger.info("S3 error occurred: Error message: {}, Error code {}",
s3Ex.getMessage(), s3Ex.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);


        logger.info(DASHES);
        logger.info("3. Download the object to another local file.");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Void> future =
s3Actions.getObjectBytesAsync(bucketName, key, savePath);
            future.join();
            logger.info("Successfully obtained bytes from S3 object and wrote to
file {}", savePath);

        } catch (RuntimeException rt) {
```

```
            Throwable cause = rt.getCause();
            if (cause instanceof S3Exception s3Ex) {
                logger.info("S3 error occurred: Error message: {}, Error code {}",
s3Ex.getMessage(), s3Ex.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("4. Perform a multipart upload.");
        waitForInputToContinue(scanner);
        String multipartKey = "multiPartKey";
        try {
            // Call the multipartUpload method
            CompletableFuture<Void> future = s3Actions.multipartUpload(bucketName,
multipartKey);
            future.join();
            logger.info("Multipart upload completed successfully for bucket '{}' and
key '{}'", bucketName, multipartKey);

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof S3Exception s3Ex) {
                logger.info("S3 error occurred: Error message: {}, Error code {}",
s3Ex.getMessage(), s3Ex.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("5. List all objects located in the Amazon S3 bucket.");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Void> future =
s3Actions.listAllObjectsAsync(bucketName);
            future.join();
```

```
            logger.info("Object listing completed successfully.");

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof S3Exception s3Ex) {
                logger.info("S3 error occurred: Error message: {}, Error code {}",
s3Ex.getMessage(), s3Ex.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("6. Copy the object to another Amazon S3 bucket.");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<String> future =
s3Actions.copyBucketObjectAsync(bucketName, key, toBucket);
            String result = future.join();
            logger.info("Copy operation result: {}", result);

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof S3Exception s3Ex) {
                logger.info("S3 error occurred: Error message: {}, Error code {}",
s3Ex.getMessage(), s3Ex.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("7. Copy the object to another Amazon S3 bucket using multi
copy.");
        waitForInputToContinue(scanner);

        try {
```

```
            CompletableFuture<String> future = s3Actions.performMultiCopy(toBucket,
bucketName, key);
            String result = future.join();
            logger.info("Copy operation result: {}", result);

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof S3Exception s3Ex) {
                logger.info("KMS error occurred: Error message: {}, Error code {}",
s3Ex.getMessage(), s3Ex.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);


        logger.info(DASHES);
        logger.info("8. Delete objects from the Amazon S3 bucket.");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Void> future =
s3Actions.deleteObjectFromBucketAsync(bucketName, key);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof S3Exception s3Ex) {
                logger.info("S3 error occurred: Error message: {}, Error code {}",
s3Ex.getMessage(), s3Ex.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            throw cause;
        }
        try {
            CompletableFuture<Void> future =
s3Actions.deleteObjectFromBucketAsync(bucketName, "multiPartKey");
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof S3Exception s3Ex) {
```

```
                logger.info("S3 error occurred: Error message: {}, Error code {}",
s3Ex.getMessage(), s3Ex.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("9. Delete the Amazon S3 bucket.");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Void> future =
s3Actions.deleteBucketAsync(bucketName);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof S3Exception s3Ex) {
                logger.info("S3 error occurred: Error message: {}, Error code {}",
s3Ex.getMessage(), s3Ex.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("You successfully completed the Amazon S3 scenario.");
        logger.info(DASHES);
    }

    private static void waitForInputToContinue(Scanner scanner) {
        while (true) {
            logger.info("");
            logger.info("Enter 'c' followed by <ENTER> to continue:");
            String input = scanner.nextLine();

            if (input.trim().equalsIgnoreCase("c")) {
                logger.info("Continuing with the program...");
```

```
                logger.info("");
                break;
            } else {
                // Handle invalid input.
                logger.info("Invalid input. Please try again.");
            }
        }
    }
}
```

Una classe wrapper che contiene le operazioni.

```
public class S3Actions {

    private static final Logger logger = LoggerFactory.getLogger(S3Actions.class);
    private static S3AsyncClient s3AsyncClient;

    public static S3AsyncClient getAsyncClient() {
        if (s3AsyncClient == null) {
            /*
            The `NettyNioAsyncHttpClient` class is part of the AWS SDK for Java,
 version 2,
            and it is designed to provide a high-performance, asynchronous HTTP
 client for interacting with AWS services.
             It uses the Netty framework to handle the underlying network
 communication and the Java NIO API to
             provide a non-blocking, event-driven approach to HTTP requests and
 responses.
             */

            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                .maxConcurrency(50)  // Adjust as needed.
                .connectionTimeout(Duration.ofSeconds(60))  // Set the connection
 timeout.
                .readTimeout(Duration.ofSeconds(60))  // Set the read timeout.
                .writeTimeout(Duration.ofSeconds(60))  // Set the write timeout.
                .build();

            ClientOverrideConfiguration overrideConfig =
 ClientOverrideConfiguration.builder()
                .apiCallTimeout(Duration.ofMinutes(2))  // Set the overall API call
 timeout.
```

```
                .apiCallAttemptTimeout(Duration.ofSeconds(90))  // Set the
individual call attempt timeout.
                .retryStrategy(RetryMode.STANDARD)
                .build();

            s3AsyncClient = S3AsyncClient.builder()
                .region(Region.US_EAST_1)
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return s3AsyncClient;
    }


    /**
     * Creates an S3 bucket asynchronously.
     *
     * @param bucketName the name of the S3 bucket to create
     * @return a {@link CompletableFuture} that completes when the bucket is created
and ready
     * @throws RuntimeException if there is a failure while creating the bucket
     */
    public CompletableFuture<Void> createBucketAsync(String bucketName) {
        CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
            .bucket(bucketName)
            .build();

        CompletableFuture<CreateBucketResponse> response =
getAsyncClient().createBucket(bucketRequest);
        return response.thenCompose(resp -> {
            S3AsyncWaiter s3Waiter = getAsyncClient().waiter();
            HeadBucketRequest bucketRequestWait = HeadBucketRequest.builder()
                .bucket(bucketName)
                .build();

            CompletableFuture<WaiterResponse<HeadBucketResponse>>
waiterResponseFuture =
                s3Waiter.waitUntilBucketExists(bucketRequestWait);
            return waiterResponseFuture.thenAccept(waiterResponse -> {
                waiterResponse.matched().response().ifPresent(headBucketResponse ->
{
                    logger.info(bucketName + " is ready");
                });
```

```
            });
        }).whenComplete((resp, ex) -> {
            if (ex != null) {
                throw new RuntimeException("Failed to create bucket", ex);
            }
        });
    }


    /**
     * Uploads a local file to an AWS S3 bucket asynchronously.
     *
     * @param bucketName the name of the S3 bucket to upload the file to
     * @param key        the key (object name) to use for the uploaded file
     * @param objectPath the local file path of the file to be uploaded
     * @return a {@link CompletableFuture} that completes with the {@link
PutObjectResponse} when the upload is successful, or throws a {@link
RuntimeException} if the upload fails
     */
    public CompletableFuture<PutObjectResponse> uploadLocalFileAsync(String
bucketName, String key, String objectPath) {
        PutObjectRequest objectRequest = PutObjectRequest.builder()
            .bucket(bucketName)
            .key(key)
            .build();

        CompletableFuture<PutObjectResponse> response =
getAsyncClient().putObject(objectRequest,
AsyncRequestBody.fromFile(Paths.get(objectPath)));
        return response.whenComplete((resp, ex) -> {
            if (ex != null) {
                throw new RuntimeException("Failed to upload file", ex);
            }
        });
    }


    /**
     * Asynchronously retrieves the bytes of an object from an Amazon S3 bucket and
writes them to a local file.
     *
     * @param bucketName the name of the S3 bucket containing the object
     * @param keyName    the key (or name) of the S3 object to retrieve
```

```
    * @param path       the local file path where the object's bytes will be
written
    * @return a {@link CompletableFuture} that completes when the object bytes have
been written to the local file
    */
   public CompletableFuture<Void> getObjectBytesAsync(String bucketName, String
keyName, String path) {
       GetObjectRequest objectRequest = GetObjectRequest.builder()
           .key(keyName)
           .bucket(bucketName)
           .build();

       CompletableFuture<ResponseBytes<GetObjectResponse>> response =
getAsyncClient().getObject(objectRequest, AsyncResponseTransformer.toBytes());
       return response.thenAccept(objectBytes -> {
           try {
               byte[] data = objectBytes.asByteArray();
               Path filePath = Paths.get(path);
               Files.write(filePath, data);
               logger.info("Successfully obtained bytes from an S3 object");
           } catch (IOException ex) {
               throw new RuntimeException("Failed to write data to file", ex);
           }
       }).whenComplete((resp, ex) -> {
           if (ex != null) {
               throw new RuntimeException("Failed to get object bytes from S3",
ex);
           }
       });
   }


   /**
    * Asynchronously lists all objects in the specified S3 bucket.
    *
    * @param bucketName the name of the S3 bucket to list objects for
    * @return a {@link CompletableFuture} that completes when all objects have been
listed
    */
   public CompletableFuture<Void> listAllObjectsAsync(String bucketName) {
       ListObjectsV2Request initialRequest = ListObjectsV2Request.builder()
           .bucket(bucketName)
           .maxKeys(1)
           .build();
```

```
        ListObjectsV2Publisher paginator =
getAsyncClient().listObjectsV2Paginator(initialRequest);
        return paginator.subscribe(response -> {
            response.contents().forEach(s3Object -> {
                logger.info("Object key: " + s3Object.key());
            });
        }).thenRun(() -> {
            logger.info("Successfully listed all objects in the bucket: " +
bucketName);
        }).exceptionally(ex -> {
            throw new RuntimeException("Failed to list objects", ex);
        });
    }


    /**
     * Asynchronously copies an object from one S3 bucket to another.
     *
     * @param fromBucket the name of the source S3 bucket
     * @param objectKey  the key (name) of the object to be copied
     * @param toBucket   the name of the destination S3 bucket
     * @return a {@link CompletableFuture} that completes with the copy result as a
{@link String}
     * @throws RuntimeException if the URL could not be encoded or an S3 exception
occurred during the copy
     */
    public CompletableFuture<String> copyBucketObjectAsync(String fromBucket, String
objectKey, String toBucket) {
        CopyObjectRequest copyReq = CopyObjectRequest.builder()
            .sourceBucket(fromBucket)
            .sourceKey(objectKey)
            .destinationBucket(toBucket)
            .destinationKey(objectKey)
            .build();

        CompletableFuture<CopyObjectResponse> response =
getAsyncClient().copyObject(copyReq);
        response.whenComplete((copyRes, ex) -> {
            if (copyRes != null) {
                logger.info("The " + objectKey + " was copied to " + toBucket);
            } else {
                throw new RuntimeException("An S3 exception occurred during copy",
ex);
```

```
            }
        });

        return response.thenApply(CopyObjectResponse::copyObjectResult)
            .thenApply(Object::toString);
    }

    /**
     * Performs a multipart upload to an Amazon S3 bucket.
     *
     * @param bucketName the name of the S3 bucket to upload the file to
     * @param key        the key (name) of the file to be uploaded
     * @return a {@link CompletableFuture} that completes when the multipart upload
is successful
     */
    public CompletableFuture<Void> multipartUpload(String bucketName, String key) {
        int mB = 1024 * 1024;

        CreateMultipartUploadRequest createMultipartUploadRequest =
CreateMultipartUploadRequest.builder()
            .bucket(bucketName)
            .key(key)
            .build();

        return getAsyncClient().createMultipartUpload(createMultipartUploadRequest)
            .thenCompose(createResponse -> {
                String uploadId = createResponse.uploadId();
                System.out.println("Upload ID: " + uploadId);

                // Upload part 1.
                UploadPartRequest uploadPartRequest1 = UploadPartRequest.builder()
                    .bucket(bucketName)
                    .key(key)
                    .uploadId(uploadId)
                    .partNumber(1)
                    .contentLength((long) (5 * mB)) // Specify the content length
                    .build();

                CompletableFuture<CompletedPart> part1Future =
getAsyncClient().uploadPart(uploadPartRequest1,
                        AsyncRequestBody.fromByteBuffer(getRandomByteBuffer(5 *
mB)))
                    .thenApply(uploadPartResponse -> CompletedPart.builder()
                        .partNumber(1)
```

```
                        .eTag(uploadPartResponse.eTag())
                        .build());

                // Upload part 2.
                UploadPartRequest uploadPartRequest2 = UploadPartRequest.builder()
                    .bucket(bucketName)
                    .key(key)
                    .uploadId(uploadId)
                    .partNumber(2)
                    .contentLength((long) (3 * mB))
                    .build();

                CompletableFuture<CompletedPart> part2Future =
getAsyncClient().uploadPart(uploadPartRequest2,
                        AsyncRequestBody.fromByteBuffer(getRandomByteBuffer(3 *
mB)))
                    .thenApply(uploadPartResponse -> CompletedPart.builder()
                        .partNumber(2)
                        .eTag(uploadPartResponse.eTag())
                        .build());

                // Combine the results of both parts.
                return CompletableFuture.allOf(part1Future, part2Future)
                    .thenCompose(v -> {
                        CompletedPart part1 = part1Future.join();
                        CompletedPart part2 = part2Future.join();

                        CompletedMultipartUpload completedMultipartUpload =
CompletedMultipartUpload.builder()
                            .parts(part1, part2)
                            .build();

                        CompleteMultipartUploadRequest
completeMultipartUploadRequest = CompleteMultipartUploadRequest.builder()
                            .bucket(bucketName)
                            .key(key)
                            .uploadId(uploadId)
                            .multipartUpload(completedMultipartUpload)
                            .build();

                        // Complete the multipart upload
                        return
getAsyncClient().completeMultipartUpload(completeMultipartUploadRequest);
                    });
```

```
            })
            .thenAccept(response -> System.out.println("Multipart upload completed
successfully"))
            .exceptionally(ex -> {
                System.err.println("Failed to complete multipart upload: " +
ex.getMessage());
                throw new RuntimeException(ex);
            });
    }


    /**
     * Deletes an object from an S3 bucket asynchronously.
     *
     * @param bucketName the name of the S3 bucket
     * @param key        the key (file name) of the object to be deleted
     * @return a {@link CompletableFuture} that completes when the object has been
deleted
     */
    public CompletableFuture<Void> deleteObjectFromBucketAsync(String bucketName,
String key) {
        DeleteObjectRequest deleteObjectRequest = DeleteObjectRequest.builder()
            .bucket(bucketName)
            .key(key)
            .build();

        CompletableFuture<DeleteObjectResponse> response =
getAsyncClient().deleteObject(deleteObjectRequest);
        response.whenComplete((deleteRes, ex) -> {
            if (deleteRes != null) {
                logger.info(key + " was deleted");
            } else {
                throw new RuntimeException("An S3 exception occurred during delete",
ex);
            }
        });

        return response.thenApply(r -> null);
    }


    /**
     * Deletes an S3 bucket asynchronously.
     *
```

```
    * @param bucket the name of the bucket to be deleted
    * @return a {@link CompletableFuture} that completes when the bucket deletion
is successful, or throws a {@link RuntimeException}
    * if an error occurs during the deletion process
    */
   public CompletableFuture<Void> deleteBucketAsync(String bucket) {
       DeleteBucketRequest deleteBucketRequest = DeleteBucketRequest.builder()
           .bucket(bucket)
           .build();

       CompletableFuture<DeleteBucketResponse> response =
getAsyncClient().deleteBucket(deleteBucketRequest);
       response.whenComplete((deleteRes, ex) -> {
           if (deleteRes != null) {
               logger.info(bucket + " was deleted.");
           } else {
               throw new RuntimeException("An S3 exception occurred during bucket
deletion", ex);
           }
       });
       return response.thenApply(r -> null);
   }

   public CompletableFuture<String> performMultiCopy(String toBucket, String
bucketName, String key) {
       CreateMultipartUploadRequest createMultipartUploadRequest =
CreateMultipartUploadRequest.builder()
           .bucket(toBucket)
           .key(key)
           .build();

       getAsyncClient().createMultipartUpload(createMultipartUploadRequest)
           .thenApply(createMultipartUploadResponse -> {
               String uploadId = createMultipartUploadResponse.uploadId();
               System.out.println("Upload ID: " + uploadId);

               UploadPartCopyRequest uploadPartCopyRequest =
UploadPartCopyRequest.builder()
                   .sourceBucket(bucketName)
                   .destinationBucket(toBucket)
                   .sourceKey(key)
                   .destinationKey(key)
                   .uploadId(uploadId)  // Use the valid uploadId.
                   .partNumber(1)  // Ensure the part number is correct.
```

```
                    .copySourceRange("bytes=0-1023")  // Adjust range as needed
                    .build();

                return getAsyncClient().uploadPartCopy(uploadPartCopyRequest);
            })
            .thenCompose(uploadPartCopyFuture -> uploadPartCopyFuture)
            .whenComplete((uploadPartCopyResponse, exception) -> {
                if (exception != null) {
                    // Handle any exceptions.
                    logger.error("Error during upload part copy: " +
 exception.getMessage());
                } else {
                    // Successfully completed the upload part copy.
                    System.out.println("Upload Part Copy completed successfully.
 ETag: " + uploadPartCopyResponse.copyPartResult().eTag());
                }
            });
        return null;
    }

    private static ByteBuffer getRandomByteBuffer(int size) {
        ByteBuffer buffer = ByteBuffer.allocate(size);
        for (int i = 0; i < size; i++) {
            buffer.put((byte) (Math.random() * 256));
        }
        buffer.flip();
        return buffer;
    }
}
```

- 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

Azioni

`AbortMultipartUpload`

Il seguente esempio di codice mostra come usare. `AbortMultipartUpload`

SDK per Java 2.x

> **Note**
>
> C'è 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.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.AbortMultipartUploadRequest;
import software.amazon.awssdk.services.s3.model.AbortMultipartUploadResponse;
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.LifecycleRule;
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.PutBucketLifecycleConfigurationResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;
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.services.sts.StsClient;
import software.amazon.awssdk.utils.builder.SdkBuilder;

import java.io.IOException;
import java.io.RandomAccessFile;
import java.net.URISyntaxException;
import java.net.URL;
```

```
import java.nio.ByteBuffer;
import java.time.Duration;
import java.time.Instant;
import java.util.ArrayList;
import java.util.Collection;
import java.util.List;
import java.util.Objects;
import java.util.UUID;

import static software.amazon.awssdk.transfer.s3.SizeConstant.KB;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class AbortMultipartUploadExamples {
    static final String bucketName = "amzn-s3-demo-bucket" + UUID.randomUUID(); //
 Change bucket name.
    static final String key = UUID.randomUUID().toString();
    static final String classPathFilePath = "/multipartUploadFiles/s3-
userguide.pdf";
    static final String filePath = getFullFilePath(classPathFilePath);
    static final S3Client s3Client = S3Client.create();
    private static final Logger logger =
 LoggerFactory.getLogger(AbortMultipartUploadExamples.class);
    private static String accountId = getAccountId();

    public static void main(String[] args) {
        doAbortIncompleteMultipartUploadsFromList();
        doAbortMultipartUploadUsingUploadId();
        doAbortIncompleteMultipartUploadsOlderThan();
        doAbortMultipartUploadsUsingLifecycleConfig();
    }

    // A wrapper method that sets up the multipart upload environment for
 abortIncompleteMultipartUploadsFromList().
    public static void doAbortIncompleteMultipartUploadsFromList() {
        createBucket();
        initiateAndInterruptMultiPartUpload("uploadThread");
```

```
        abortIncompleteMultipartUploadsFromList();
        deleteResources();
    }

    /**
     * Aborts all incomplete multipart uploads from the specified S3 bucket.
     * <p>
     * This method retrieves a list of all incomplete multipart uploads in the
specified S3 bucket,
     * and then aborts each of those uploads.
     */
    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);
            }
        }
    }

    // A wrapper method that sets up the multipart upload environment for
abortIncompleteMultipartUploadsOlderThan().
    static void doAbortIncompleteMultipartUploadsOlderThan() {
        createBucket();
        Instant secondUploadInstant = initiateAndInterruptTwoUploads();
```

```
        abortIncompleteMultipartUploadsOlderThan(secondUploadInstant);
        deleteResources();
    }

    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);
                }
            }
        }
    }

    // A wrapper method that sets up the multipart upload environment for
abortMultipartUploadUsingUploadId().
    static void doAbortMultipartUploadUsingUploadId() {
        createBucket();
        try {
            abortMultipartUploadUsingUploadId();
        } catch (S3Exception e) {
            logger.error(e.getMessage());
```

```
        } finally {
            deleteResources();
        }
    }

    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);
        }
    }

    // A wrapper method that sets up the multipart upload environment for
abortMultipartUploadsUsingLifecycleConfig().
    static void doAbortMultipartUploadsUsingLifecycleConfig() {
        createBucket();
        try {
            abortMultipartUploadsUsingLifecycleConfig();
        } catch (S3Exception e) {
            logger.error(e.getMessage());
        } finally {
            deleteResources();
        }
    }

    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());
        }
    }

    /************************
     Multipart upload methods
     ************************/

    static void initiateAndInterruptMultiPartUpload(String threadName) {
        Runnable upload = () -> {
            try {
                AbortMultipartUploadExamples.doMultipartUpload();
            } catch (SdkException e) {
                logger.error(e.getMessage());
            }
        };
        Thread uploadThread = new Thread(upload, threadName);
        uploadThread.start();
        try {
            Thread.sleep(Duration.ofSeconds(1).toMillis()); // Give the multipart
upload time to register.
        } catch (InterruptedException e) {
            logger.error(e.getMessage());
        }
        uploadThread.interrupt();
    }

    static Instant initiateAndInterruptTwoUploads() {
        Instant firstUploadInstant = Instant.now();
        initiateAndInterruptMultiPartUpload("uploadThread1");
        try {
            Thread.sleep(Duration.ofSeconds(5).toMillis());
        } catch (InterruptedException e) {
            logger.error(e.getMessage());
        }
        Instant secondUploadInstant = Instant.now();
        initiateAndInterruptMultiPartUpload("uploadThread2");
```

```
        return secondUploadInstant;
    }

    static void doMultipartUpload() {
        String uploadId = step1CreateMultipartUpload();
        List<CompletedPart> completedParts = step2UploadParts(uploadId);
        step3CompleteMultipartUpload(uploadId, completedParts);
    }

    static String step1CreateMultipartUpload() {
        CreateMultipartUploadResponse createMultipartUploadResponse =
s3Client.createMultipartUpload(b -> b
            .bucket(bucketName)
            .key(key));
        return createMultipartUploadResponse.uploadId();
    }

    static List<CompletedPart> step2UploadParts(String uploadId) {
        int partNumber = 1;
        List<CompletedPart> completedParts = new ArrayList<>();
        ByteBuffer bb = ByteBuffer.allocate(Long.valueOf(1024 * KB).intValue());

        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);
                logger.info("Part {} upload", partNumber);

                bb.clear();
                position += read;
                partNumber++;
            }
        } catch (IOException | S3Exception e) {
            logger.error(e.getMessage());
            return null;
        }
        return completedParts;
    }

    static void step3CompleteMultipartUpload(String uploadId, List<CompletedPart>
completedParts) {
        s3Client.completeMultipartUpload(b -> b
            .bucket(bucketName)
            .key(key)
            .uploadId(uploadId)

.multipartUpload(CompletedMultipartUpload.builder().parts(completedParts).build()));
    }

    static String startUploadReturningUploadId() {
        String uploadId = step1CreateMultipartUpload();
        doMultipartUploadWithUploadId(uploadId);
        return uploadId;

    }

    static void doMultipartUploadWithUploadId(String uploadId) {
        new Thread(() -> {
            try {
                List<CompletedPart> completedParts = step2UploadParts(uploadId);
                step3CompleteMultipartUpload(uploadId, completedParts);
            } catch (SdkException e) {
                logger.error(e.getMessage());
            }
        }, "upload thread").start();
        try {
            Thread.sleep(Duration.ofSeconds(2L).toMillis());
```

```
        } catch (InterruptedException e) {
            logger.error(e.getMessage());
            System.exit(1);
        }
    }

    /************************
     Resource handling methods
     ************************/

    static void createBucket() {
        logger.info("Creating bucket: [{}]", bucketName);
        s3Client.createBucket(b -> b.bucket(bucketName));
        try (S3Waiter s3Waiter = s3Client.waiter()) {
            s3Waiter.waitUntilBucketExists(b -> b.bucket(bucketName));
        }
        logger.info("Bucket created.");
    }

    static void deleteResources() {
        logger.info("Deleting resources ...");
        s3Client.deleteObject(b -> b.bucket(bucketName).key(key));
        s3Client.deleteBucket(b -> b.bucket(bucketName));
        try (S3Waiter s3Waiter = s3Client.waiter()) {
            s3Waiter.waitUntilBucketNotExists(b -> b.bucket(bucketName));
        }
        logger.info("Resources deleted.");
    }

    private static String getAccountId() {
        try (StsClient stsClient = StsClient.create()) {
            return stsClient.getCallerIdentity().account();
        }
    }

    static String getFullFilePath(String filePath) {
        URL uploadDirectoryURL = PerformMultiPartUpload.class.getResource(filePath);
        String fullFilePath;
        try {
            fullFilePath =
Objects.requireNonNull(uploadDirectoryURL).toURI().getPath();
        } catch (URISyntaxException e) {
            throw new RuntimeException(e);
        }
```

```
        return fullFilePath;
    }
}
```

- Per i dettagli sull'API, AbortMultipartUploadconsulta AWS SDK for Java 2.x API Reference.

`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.

```
   /**
    * Asynchronously copies an object from one S3 bucket to another.
    *
    * @param fromBucket the name of the source S3 bucket
    * @param objectKey  the key (name) of the object to be copied
    * @param toBucket   the name of the destination S3 bucket
    * @return a {@link CompletableFuture} that completes with the copy result as a
{@link String}
    * @throws RuntimeException if the URL could not be encoded or an S3 exception
occurred during the copy
    */
   public CompletableFuture<String> copyBucketObjectAsync(String fromBucket, String
objectKey, String toBucket) {
       CopyObjectRequest copyReq = CopyObjectRequest.builder()
           .sourceBucket(fromBucket)
           .sourceKey(objectKey)
           .destinationBucket(toBucket)
           .destinationKey(objectKey)
           .build();
```

```
        CompletableFuture<CopyObjectResponse> response =
getAsyncClient().copyObject(copyReq);
        response.whenComplete((copyRes, ex) -> {
            if (copyRes != null) {
                logger.info("The " + objectKey + " was copied to " + toBucket);
            } else {
                throw new RuntimeException("An S3 exception occurred during copy",
ex);
            }
        });

        return response.thenApply(CopyObjectResponse::copyObjectResult)
            .thenApply(Object::toString);
    }
```

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.

```
    /**
     * Creates an S3 bucket asynchronously.
     *
     * @param bucketName the name of the S3 bucket to create
     * @return a {@link CompletableFuture} that completes when the bucket is created
and ready
     * @throws RuntimeException if there is a failure while creating the bucket
     */
    public CompletableFuture<Void> createBucketAsync(String bucketName) {
        CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
            .bucket(bucketName)
            .build();

        CompletableFuture<CreateBucketResponse> response =
getAsyncClient().createBucket(bucketRequest);
        return response.thenCompose(resp -> {
            S3AsyncWaiter s3Waiter = getAsyncClient().waiter();
            HeadBucketRequest bucketRequestWait = HeadBucketRequest.builder()
```

```
                .bucket(bucketName)
                .build();

            CompletableFuture<WaiterResponse<HeadBucketResponse>>
waiterResponseFuture =
                s3Waiter.waitUntilBucketExists(bucketRequestWait);
            return waiterResponseFuture.thenAccept(waiterResponse -> {
                waiterResponse.matched().response().ifPresent(headBucketResponse ->
{
                    logger.info(bucketName + " is ready");
                });
            });
        }).whenComplete((resp, ex) -> {
            if (ex != null) {
                throw new RuntimeException("Failed to create bucket", ex);
            }
        });
    }
```

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.xAPI 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.

```
   /**
    * Deletes an S3 bucket asynchronously.
    *
    * @param bucket the name of the bucket to be deleted
    * @return a {@link CompletableFuture} that completes when the bucket deletion
is successful, or throws a {@link RuntimeException}
    * if an error occurs during the deletion process
    */
   public CompletableFuture<Void> deleteBucketAsync(String bucket) {
       DeleteBucketRequest deleteBucketRequest = DeleteBucketRequest.builder()
           .bucket(bucket)
           .build();

       CompletableFuture<DeleteBucketResponse> response =
getAsyncClient().deleteBucket(deleteBucketRequest);
       response.whenComplete((deleteRes, ex) -> {
           if (deleteRes != null) {
               logger.info(bucket + " was deleted.");
           } else {
               throw new RuntimeException("An S3 exception occurred during bucket
deletion", ex);
           }
       });
       return response.thenApply(r -> null);
   }
```

- Per i dettagli sull'API, DeleteBucketconsulta 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();
    }

    /**
     * Deletes the S3 bucket policy for the specified bucket.
     *
     * @param s3 the {@link S3Client} instance to use for the operation
     * @param bucketName the name of the S3 bucket for which the policy should be
 deleted
     *
     * @throws S3Exception if there is an error deleting 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, DeleteBucketPolicyconsulta 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.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * 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();
    }

    /**
     * Deletes the website configuration for an Amazon S3 bucket.
     *
     * @param s3 The {@link S3Client} instance used to interact with Amazon S3.
     * @param bucketName The name of the S3 bucket for which the website
 configuration should be deleted.
     * @throws S3Exception If an error occurs while deleting the website
 configuration.
     */
    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, DeleteBucketWebsiteconsulta 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.

```
   /**
    * Deletes an object from an S3 bucket asynchronously.
    *
    * @param bucketName the name of the S3 bucket
    * @param key        the key (file name) of the object to be deleted
    * @return a {@link CompletableFuture} that completes when the object has been
deleted
    */
   public CompletableFuture<Void> deleteObjectFromBucketAsync(String bucketName,
String key) {
       DeleteObjectRequest deleteObjectRequest = DeleteObjectRequest.builder()
           .bucket(bucketName)
           .key(key)
           .build();

       CompletableFuture<DeleteObjectResponse> response =
getAsyncClient().deleteObject(deleteObjectRequest);
       response.whenComplete((deleteRes, ex) -> {
           if (deleteRes != null) {
               logger.info(key + " was deleted");
           } else {
               throw new RuntimeException("An S3 exception occurred during delete",
ex);
           }
       });

       return response.thenApply(r -> null);
   }
```

- Per i dettagli sull'API, DeleteObjectconsulta 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.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * 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();
    }

    /**
     * Deletes multiple objects from an Amazon S3 bucket.
     *
     * @param s3 An Amazon S3 client object.
     * @param bucketName The name of the Amazon S3 bucket to delete objects from.
     */
    public static void deleteBucketObjects(S3Client s3, String bucketName) {
        // Upload three sample objects to the specfied Amazon S3 bucket.
        ArrayList<ObjectIdentifier> keys = new ArrayList<>();
        PutObjectRequest putOb;
        ObjectIdentifier objectId;

        for (int i = 0; i < 3; i++) {
            String keyName = "delete object example " + i;
            objectId = ObjectIdentifier.builder()
                .key(keyName)
                .build();

            putOb = PutObjectRequest.builder()
                .bucket(bucketName)
                .key(keyName)
                .build();

            s3.putObject(putOb, RequestBody.fromString(keyName));
            keys.add(objectId);
        }

        System.out.println(keys.size() + " objects successfully created.");
```

```
        // Delete multiple objects in one request.
        Delete del = Delete.builder()
            .objects(keys)
            .build();

        try {
            DeleteObjectsRequest multiObjectDeleteRequest =
 DeleteObjectsRequest.builder()
                .bucket(bucketName)
                .delete(del)
                .build();

            s3.deleteObjects(multiObjectDeleteRequest);
            System.out.println("Multiple objects are deleted!");

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Per i dettagli sull'API, DeleteObjectsconsulta 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.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * 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!");
```

```
    }

    /**
     * Retrieves the Access Control List (ACL) for an object in an Amazon S3 bucket.
     *
     * @param s3 The S3Client object used to interact with the Amazon S3 service.
     * @param objectKey The key of the object for which the ACL is to be retrieved.
     * @param bucketName The name of the bucket containing the object.
     * @return The ID of the grantee who has permission on the object, or an empty
 string if an error occurs.
     */
    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, GetBucketAclconsulta 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.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * 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();
    }

    /**
     * Retrieves the policy for the specified Amazon S3 bucket.
     *
     * @param s3 the {@link S3Client} instance to use for making the request
     * @param bucketName the name of the S3 bucket for which to retrieve the policy
     * @return the policy text for the specified bucket, or an empty string if an
 error occurs
     */
    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, GetBucketPolicyconsulta AWS SDK for Java 2.x API Reference.

`GetBucketReplication`

Il seguente esempio di codice mostra come utilizzare`GetBucketReplication`.

SDK per Java 2.x

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
   /**
    * Retrieves the replication details for the specified S3 bucket.
    *
    * @param s3Client           the S3 client used to interact with the S3 service
    * @param sourceBucketName   the name of the S3 bucket to retrieve the
replication details for
    *
    * @throws S3Exception if there is an error retrieving the replication details
    */
   public static void getReplicationDetails(S3Client s3Client, String
sourceBucketName) {
       GetBucketReplicationRequest getRequest =
GetBucketReplicationRequest.builder()
           .bucket(sourceBucketName)
           .build();

       try {
           ReplicationConfiguration replicationConfig =
s3Client.getBucketReplication(getRequest).replicationConfiguration();
           ReplicationRule rule = replicationConfig.rules().get(0);
           System.out.println("Retrieved destination bucket: " +
rule.destination().bucket());
           System.out.println("Retrieved priority: " + rule.priority());
           System.out.println("Retrieved source-bucket replication rule status: " +
rule.status());

       } catch (S3Exception e) {
           System.err.println("Failed to retrieve replication details: " +
e.awsErrorDetails().errorMessage());
       }
   }
```

- Per i dettagli sull'API, GetBucketReplicationconsulta 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.

```
   /**
    * Asynchronously retrieves the bytes of an object from an Amazon S3 bucket and
writes them to a local file.
    *
    * @param bucketName the name of the S3 bucket containing the object
    * @param keyName    the key (or name) of the S3 object to retrieve
    * @param path       the local file path where the object's bytes will be
written
    * @return a {@link CompletableFuture} that completes when the object bytes have
been written to the local file
    */
   public CompletableFuture<Void> getObjectBytesAsync(String bucketName, String
keyName, String path) {
       GetObjectRequest objectRequest = GetObjectRequest.builder()
           .key(keyName)
           .bucket(bucketName)
           .build();

       CompletableFuture<ResponseBytes<GetObjectResponse>> response =
getAsyncClient().getObject(objectRequest, AsyncResponseTransformer.toBytes());
       return response.thenAccept(objectBytes -> {
           try {
               byte[] data = objectBytes.asByteArray();
```

```
                Path filePath = Paths.get(path);
                Files.write(filePath, data);
                logger.info("Successfully obtained bytes from an S3 object");
            } catch (IOException ex) {
                throw new RuntimeException("Failed to write data to file", ex);
            }
        }).whenComplete((resp, ex) -> {
            if (ex != null) {
                throw new RuntimeException("Failed to get object bytes from S3",
ex);
            }
        });
    }
```

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().contentLength());
        return downloadResult.response().contentLength();
    }
```

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.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * 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();
    }

    /**
     * Lists the tags associated with an Amazon S3 object.
     *
     * @param s3 the S3Client object used to interact with the Amazon S3 service
     * @param bucketName the name of the S3 bucket that contains the object
     * @param keyName the key (name) of the S3 object
     */
    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.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * 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();
    }

    /**
     * Retrieves the URL for a specific object in an Amazon S3 bucket.
     *
     * @param s3 the S3Client object used to interact with the Amazon S3 service
     * @param bucketName the name of the S3 bucket where the object is stored
     * @param keyName the name of the object for which the URL should be retrieved
     * @throws S3Exception if there is an error retrieving the URL for the specified
 object
     */
    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.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * 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();
    }
```

```
   /**
    * Generates a pre-signed URL for an Amazon S3 object.
    *
    * @param presigner The {@link S3Presigner} instance to use for generating the
pre-signed URL.
    * @param bucketName The name of the Amazon S3 bucket where the object is
stored.
    * @param keyName The key name (file name) of the object in the Amazon S3
bucket.
    *
    * @throws S3Exception If there is an error interacting with the Amazon S3
service.
    * @throws IOException If there is an error opening the HTTP connection or
reading/writing the request/response.
    */
   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.getStackTrace();
        }
    }
}
```

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.
 * <p>
```

```
 * For more information, see the following documentation topic:
 * <p>
 * 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);
        s3.close();
    }

    /**
     * Retrieves the bytes of an object stored in an Amazon S3 bucket and saves them
 to a local file.
     *
     * @param s3 The S3Client instance used to interact with the Amazon S3 service.
     * @param bucketName The name of the S3 bucket where the object is stored.
     * @param keyName The key (or name) of the S3 object.
     * @param path The local file path where the object's bytes will be saved.
```

```
     * @throws IOException If an I/O error occurs while writing the bytes to the
 local file.
     * @throws S3Exception If an error occurs while retrieving the object from the
 S3 bucket.
     */
    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 GetObjectin 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, GetObjectLegalHoldconsulta 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.objectLockConfiguration().objectLockEnabled());
       System.out.println("\tRule: "+
response.objectLockConfiguration().rule().defaultRetention());
   }
```

- Per i dettagli sull'API, GetObjectLockConfigurationconsulta 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("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;
       }
   }
```

- Per i dettagli sull'API, GetObjectRetentionconsulta 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.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * 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();
    }

    /**
     * Retrieves the content type of an object stored in an Amazon S3 bucket.
     *
```

```
     * @param s3 an instance of the {@link S3Client} class, which is used to
 interact with the Amazon S3 service
     * @param bucketName the name of the S3 bucket where the object is stored
     * @param keyName the key (file name) of the object in the S3 bucket
     */
    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();
    }

    /**
     * Checks the restoration status of an Amazon S3 object.
     *
     * @param s3         an instance of the {@link S3Client} class used to interact
with the Amazon S3 service
     * @param bucketName the name of the Amazon S3 bucket where the object is stored
     * @param keyName    the name of the Amazon S3 object to be checked
     * @throws S3Exception if an error occurs while interacting with the Amazon S3
service
     */
    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, HeadObjectconsulta 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.paginators.ListBucketsIterable;
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * 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);

    }

    /**
     * Lists all the S3 buckets available in the current AWS account.
```

```
     *
     * @param s3 The {@link S3Client} instance to use for interacting with the
 Amazon S3 service.
     */
    public static void listAllBuckets(S3Client s3) {
        ListBucketsIterable response = s3.listBucketsPaginator();
        response.buckets().forEach(bucket ->
            System.out.println("Bucket Name: " + bucket.name()));
    }
```

- Per i dettagli sull'API, ListBucketsconsulta 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();
    }

    /**
     * Lists the multipart uploads currently in progress in the specified Amazon S3
 bucket.
     *
     * @param s3 the S3Client object used to interact with Amazon S3
     * @param bucketName the name of the Amazon S3 bucket to list the multipart
 uploads for
     */
    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, ListMultipartUploadsconsulta 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.

```
    /**
     * Asynchronously lists all objects in the specified S3 bucket.
     *
     * @param bucketName the name of the S3 bucket to list objects for
     * @return a {@link CompletableFuture} that completes when all objects have been
 listed
     */
    public CompletableFuture<Void> listAllObjectsAsync(String bucketName) {
        ListObjectsV2Request initialRequest = ListObjectsV2Request.builder()
            .bucket(bucketName)
            .maxKeys(1)
            .build();
```

```
        ListObjectsV2Publisher paginator =
 getAsyncClient().listObjectsV2Paginator(initialRequest);
        return paginator.subscribe(response -> {
            response.contents().forEach(s3Object -> {
                logger.info("Object key: " + s3Object.key());
            });
        }).thenRun(() -> {
            logger.info("Successfully listed all objects in the bucket: " +
 bucketName);
        }).exceptionally(ex -> {
            throw new RuntimeException("Failed to list objects", ex);
        });
    }
```

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();
    }

    /**
     * Lists the objects in the specified S3 bucket.
     *
     * @param s3 the S3Client instance used to interact with Amazon S3
     * @param bucketName the name of the S3 bucket to list the objects from
     */
    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 utilizzare`PutBucketAcl`.

SDK per Java 2.x

> **Note**
>
> C'è 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.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * 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);
            return;
        }

        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();
    }

    /**
     * Sets the Access Control List (ACL) for an Amazon S3 bucket.
     *
     * @param s3 the S3Client instance to be used for the operation
     * @param bucketName the name of the S3 bucket to set the ACL for
     * @param id the ID of the AWS user or account that will be granted full control
of the bucket
     * @throws S3Exception if an error occurs while setting the bucket ACL
     */
    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, PutBucketAclconsulta 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.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * 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();
    }

    /**
     * Deletes the CORS (Cross-Origin Resource Sharing) configuration for an Amazon
 S3 bucket.
     *
```

```
    * @param s3            the {@link S3Client} instance used to interact with the
Amazon S3 service
    * @param bucketName    the name of the Amazon S3 bucket for which the CORS
configuration should be deleted
    * @param accountId     the expected AWS account ID of the bucket owner
    *
    * @throws S3Exception if an error occurs while deleting the CORS configuration
for the bucket
    */
   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);
       }
   }

   /**
    * Retrieves the CORS (Cross-Origin Resource Sharing) configuration for the
specified S3 bucket.
    *
    * @param s3 the S3Client instance to use for the operation
    * @param bucketName the name of the S3 bucket to retrieve the CORS
configuration for
    * @param accountId the expected bucket owner's account ID
    *
    * @throws S3Exception if there is an error retrieving the CORS configuration
    */
   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);
        }
    }

    /**
     * Sets the Cross-Origin Resource Sharing (CORS) rules for an Amazon S3 bucket.
     *
     * @param s3 The S3Client object used to interact with the Amazon S3 service.
     * @param bucketName The name of the S3 bucket to set the CORS rules for.
     * @param accountId The AWS account ID of the bucket owner.
     */
    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, PutBucketCorsconsulta 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.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * 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();
    }

    /**
     * Sets the lifecycle configuration for an Amazon S3 bucket.
     *
     * @param s3           The Amazon S3 client to use for the operation.
     * @param bucketName   The name of the Amazon S3 bucket.
     * @param accountId    The expected owner of the Amazon S3 bucket.
     *
     * @throws S3Exception if there is an error setting the lifecycle configuration.
     */
    public static void setLifecycleConfig(S3Client s3, String bucketName, String
accountId) {
        try {
            // Create a rule to archive objects with the "glacierobjects/" prefix to
the
            // S3 Glacier Flexible Retrieval storage class immediately.
            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);
        }
    }

    /**
```

```
    * Retrieves the lifecycle configuration for an Amazon S3 bucket and adds a new
lifecycle rule to it.
    *
    * @param s3 the S3Client instance used to interact with Amazon S3
    * @param bucketName the name of the Amazon S3 bucket
    * @param accountId the expected owner of the Amazon S3 bucket
    */
   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);
        }
    }

    /**
     * Deletes the lifecycle configuration for an Amazon S3 bucket.
     *
     * @param s3 the {@link S3Client} to use for the operation
     * @param bucketName the name of the S3 bucket
     * @param accountId the expected account owner of the S3 bucket
     *
     * @throws S3Exception if an error occurs while deleting the lifecycle
configuration
     */
    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, PutBucketLifecycleConfigurationconsulta 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.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * 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();
    }

    /**
     * Sets the policy for an Amazon S3 bucket.
     *
     * @param s3         the {@link S3Client} object used to interact with the
 Amazon S3 service
     * @param bucketName the name of the Amazon S3 bucket
     * @param policyText the text of the policy to be set on the bucket
     * @throws S3Exception if there is an error setting the bucket policy
```

```
     */
    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!");
    }

    /**
     * Retrieves the bucket policy from a specified file.
     *
     * @param policyFile the path to the file containing the bucket policy
     * @return the content of the bucket policy file as a string
     */
    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, PutBucketPolicyconsulta AWS SDK for Java 2.x API Reference.

`PutBucketReplication`

Il seguente esempio di codice mostra come utilizzare`PutBucketReplication`.

SDK per Java 2.x

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
    /**
     * Sets the replication configuration for an Amazon S3 bucket.
     *
     * @param s3Client             the S3Client instance to use for the operation
     * @param sourceBucketName     the name of the source bucket
     * @param destBucketName       the name of the destination bucket
     * @param destinationBucketARN the Amazon Resource Name (ARN) of the destination
 bucket
     * @param roleARN              the ARN of the IAM role to use for the
 replication configuration
     */
    public static void setReplication(S3Client s3Client, String sourceBucketName,
 String destBucketName, String destinationBucketARN, String roleARN) {
```

```
        try {
            Destination destination = Destination.builder()
                .bucket(destinationBucketARN)
                .storageClass(StorageClass.STANDARD)
                .build();

            // Define a prefix filter for replication.
            ReplicationRuleFilter ruleFilter = ReplicationRuleFilter.builder()
                .prefix("documents/")
                .build();

            // Define delete marker replication setting.
            DeleteMarkerReplication deleteMarkerReplication =
DeleteMarkerReplication.builder()
                .status(DeleteMarkerReplicationStatus.DISABLED)
                .build();

            // Create the replication rule.
            ReplicationRule replicationRule = ReplicationRule.builder()
                .priority(1)
                .filter(ruleFilter)
                .status(ReplicationRuleStatus.ENABLED)
                .deleteMarkerReplication(deleteMarkerReplication)
                .destination(destination)
                .build();

            List<ReplicationRule> replicationRuleList = new ArrayList<>();
            replicationRuleList.add(replicationRule);

            // Define the replication configuration with IAM role.
            ReplicationConfiguration configuration =
ReplicationConfiguration.builder()
                .role(roleARN)
                .rules(replicationRuleList)
                .build();

            // Apply the replication configuration to the source bucket.
            PutBucketReplicationRequest replicationRequest =
PutBucketReplicationRequest.builder()
                .bucket(sourceBucketName)
                .replicationConfiguration(configuration)
                .build();

            s3Client.putBucketReplication(replicationRequest);
```

```
            System.out.println("Replication configuration set successfully.");

        } catch (IllegalArgumentException e) {
            System.err.println("Configuration error: " + e.getMessage());
        } catch (S3Exception e) {
            System.err.println("S3 Exception: " +
e.awsErrorDetails().errorMessage());
            System.err.println("Status Code: " + e.statusCode());
            System.err.println("Error Code: " + e.awsErrorDetails().errorCode());


        } catch (SdkException e) {
            System.err.println("SDK Exception: " + e.getMessage());
        }
    }
```

- Per i dettagli sull'API, PutBucketReplicationconsulta AWS SDK for Java 2.x API Reference.

`PutBucketVersioning`

Il seguente esempio di codice mostra come utilizzare`PutBucketVersioning`.

SDK per Java 2.x

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
    /**
     * Enables bucket versioning for the specified S3 bucket.
     *
     * @param s3Client the S3 client to use for the operation
     * @param bucketName the name of the S3 bucket to enable versioning for
     */
    public static void enableBucketVersioning(S3Client s3Client, String bucketName){
        VersioningConfiguration versioningConfiguration =
VersioningConfiguration.builder()
            .status(BucketVersioningStatus.ENABLED)
            .build();
```

```
        PutBucketVersioningRequest versioningRequest =
 PutBucketVersioningRequest.builder()
             .bucket(bucketName)
             .versioningConfiguration(versioningConfiguration)
             .build();

        s3Client.putBucketVersioning(versioningRequest);
        System.out.println("Bucket versioning has been enabled for "+bucketName);
    }
```

- Per i dettagli sull'API, PutBucketVersioningconsulta 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.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * 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();
    }

    /**
     * Sets the website configuration for an Amazon S3 bucket.
     *
     * @param s3 The {@link S3Client} instance to use for the AWS SDK operations.
     * @param bucketName The name of the S3 bucket to configure.
     * @param indexDoc The name of the index document to use for the website
 configuration.
     */
    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, PutBucketWebsiteconsulta 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.

```
    /**
     * Uploads a local file to an AWS S3 bucket asynchronously.
     *
     * @param bucketName the name of the S3 bucket to upload the file to
     * @param key        the key (object name) to use for the uploaded file
     * @param objectPath the local file path of the file to be uploaded
```

```
     * @return a {@link CompletableFuture} that completes with the {@link
 PutObjectResponse} when the upload is successful, or throws a {@link
 RuntimeException} if the upload fails
     */
    public CompletableFuture<PutObjectResponse> uploadLocalFileAsync(String
 bucketName, String key, String objectPath) {
        PutObjectRequest objectRequest = PutObjectRequest.builder()
            .bucket(bucketName)
            .key(key)
            .build();

        CompletableFuture<PutObjectResponse> response =
 getAsyncClient().putObject(objectRequest,
 AsyncRequestBody.fromFile(Paths.get(objectPath)));
        return response.whenComplete((resp, ex) -> {
            if (ex != null) {
                throw new RuntimeException("Failed to upload file", ex);
            }
        });
    }
```

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.

```
    /**
     * Puts tags on an Amazon S3 object.
     *
     * @param s3 An {@link S3Client} object that represents the Amazon S3 client.
     * @param bucketName The name of the Amazon S3 bucket.
     * @param objectKey The key of the Amazon S3 object.
     * @param objectPath The file path of the object to be uploaded.
     */
    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);
        }
    }

    /**
     * Updates the tags associated with an object in an Amazon S3 bucket.
     *
     * @param s3 an instance of the S3Client class, which is used to interact with
the Amazon S3 service
     * @param bucketName the name of the S3 bucket containing the object
     * @param objectKey the key (or name) of the object in the S3 bucket
     * @throws S3Exception if there is an error updating the object's tags
     */
    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);
        }
    }

    /**
     * Retrieves the contents of a file as a byte array.
     *
     * @param filePath the path of the file to be read
     * @return a byte array containing the contents of the file, or null if an error
occurs
     */
    private static byte[] getObjectFile(String filePath) {
        FileInputStream fileInputStream = null;
        byte[] bytesArray = null;
```

```
        try {
            File file = new File(filePath);
            bytesArray = new byte[(int) file.length()];
            fileInputStream = new FileInputStream(file);
            fileInputStream.read(bytesArray);

        } catch (IOException e) {
            e.printStackTrace();
        } finally {
            if (fileInputStream != null) {
                try {
                    fileInputStream.close();
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }
        }

        return bytesArray;
    }
}
```

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.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
```

```
 * 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();
    }

    /**
     * Uploads an object to an Amazon S3 bucket with metadata.
     *
     * @param s3 the S3Client object used to interact with the Amazon S3 service
     * @param bucketName the name of the S3 bucket to upload the object to
     * @param objectKey the name of the object to be uploaded
     * @param objectPath the local file path of the object to be uploaded
     */
```

```
    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.
```

```
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * 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();

        setRentionPeriod(s3, key, bucketName);
        s3.close();
    }

    /**
     * Sets the retention period for an object in an Amazon S3 bucket.
     *
     * @param s3     the S3Client object used to interact with the Amazon S3 service
     * @param key    the key (name) of the object in the S3 bucket
     * @param bucket the name of the S3 bucket where the object is stored
     *
     * @throws S3Exception if an error occurs while setting the object retention
 period
```

```
     */
    public static void setRentionPeriod(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, PutObjectLegalHoldconsulta 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 rention = DefaultRetention.builder()
           .days(1)
           .mode(ObjectLockRetentionMode.GOVERNANCE)
           .build();

       ObjectLockRule lockRule = ObjectLockRule.builder()
           .defaultRetention(rention)
           .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.xAPI 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, PutObjectRetentionconsulta 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();
    }

    /**
     * Restores an S3 object from the Glacier storage class.
     *
     * @param s3                  an instance of the {@link S3Client} to be used
 for interacting with Amazon S3
     * @param bucketName          the name of the S3 bucket where the object is
 stored
```

```
     * @param keyName              the key (object name) of the S3 object to be
 restored
     * @param expectedBucketOwner  the AWS account ID of the expected bucket owner
     */
    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, RestoreObjectconsulta 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 = "amzn-s3-demo-bucket-" + 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:
{}\nbytesProcessed: {}\nbytesReturned:{}",
                            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.

`UploadPartCopy`

Il seguente esempio di codice mostra come utilizzare`UploadPartCopy`.

SDK per Java 2.x

> **Note**
>
> C'è 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 CompletableFuture<String> performMultiCopy(String toBucket, String
bucketName, String key) {
        CreateMultipartUploadRequest createMultipartUploadRequest =
CreateMultipartUploadRequest.builder()
            .bucket(toBucket)
            .key(key)
            .build();

        getAsyncClient().createMultipartUpload(createMultipartUploadRequest)
            .thenApply(createMultipartUploadResponse -> {
                String uploadId = createMultipartUploadResponse.uploadId();
                System.out.println("Upload ID: " + uploadId);

                UploadPartCopyRequest uploadPartCopyRequest =
UploadPartCopyRequest.builder()
```

```
                    .sourceBucket(bucketName)
                    .destinationBucket(toBucket)
                    .sourceKey(key)
                    .destinationKey(key)
                    .uploadId(uploadId)  // Use the valid uploadId.
                    .partNumber(1)  // Ensure the part number is correct.
                    .copySourceRange("bytes=0-1023")  // Adjust range as needed
                    .build();

                return getAsyncClient().uploadPartCopy(uploadPartCopyRequest);
            })
            .thenCompose(uploadPartCopyFuture -> uploadPartCopyFuture)
            .whenComplete((uploadPartCopyResponse, exception) -> {
                if (exception != null) {
                    // Handle any exceptions.
                    logger.error("Error during upload part copy: " +
exception.getMessage());
                } else {
                    // Successfully completed the upload part copy.
                    System.out.println("Upload Part Copy completed successfully.
ETag: " + uploadPartCopyResponse.copyPartResult().eTag());
                }
            });
        return null;
    }
```

- Per i dettagli sull'API, UploadPartCopyconsulta AWS SDK for Java 2.x API Reference.

Scenari

Controlla se esiste un bucket

Il seguente esempio di codice mostra come verificare se esiste un bucket.

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.

Puoi utilizzare il seguente `doesBucketExists` metodo in sostituzione del metodo SDK for Java V1 doesBucketExist AmazonS3Client# V2 (String).

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.awscore.exception.AwsServiceException;
import software.amazon.awssdk.http.HttpStatusCode;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.utils.Validate;

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

    public static void main(String[] args) {
        DoesBucketExist doesBucketExist = new DoesBucketExist();

        final S3Client s3SyncClient = S3Client.builder().build();
        final String bucketName = "amzn-s3-demo-bucket"; // Change to the bucket
 name that you want to check.

        boolean exists = doesBucketExist.doesBucketExist(bucketName, s3SyncClient);
        logger.info("Bucket exists: {}", exists);
    }

    /**
     * Checks if the specified bucket exists. Amazon S3 buckets are named in a
 global namespace; use this method to
     * determine if a specified bucket name already exists, and therefore can't be
 used to create a new bucket.
     * <p>
     * Internally this method uses the <a
     * href="https://sdk.amazonaws.com/java/api/latest/software/amazon/awssdk/
services/s3/
S3Client.html#getBucketAcl(java.util.function.Consumer)">S3Client.getBucketAcl(String)</
a>
     * operation to determine whether the bucket exists.
     * <p>
     * This method is equivalent to the AWS SDK for Java V1's <a
     * href="https://docs.aws.amazon.com/AWSJavaSDK/latest/javadoc/
com/amazonaws/services/s3/AmazonS3Client.html#doesBucketExistV2-
java.lang.String-">AmazonS3Client#doesBucketExistV2(String)</a>.
```

```
     *
     * @param bucketName   The name of the bucket to check.
     * @param s3SyncClient An <code>S3Client</code> instance. The method checks for
 the bucket in the AWS Region
     *                     configured on the instance.
     * @return The value true if the specified bucket exists in Amazon S3; the value
 false if there is no bucket in
     *         Amazon S3 with that name.
     */
    public boolean doesBucketExist(String bucketName, S3Client s3SyncClient) {
        try {
            Validate.notEmpty(bucketName, "The bucket name must not be null or an
 empty string.", "");
            s3SyncClient.getBucketAcl(r -> r.bucket(bucketName));
            return true;
        } catch (AwsServiceException ase) {
            // A redirect error or an AccessDenied exception means the bucket exists
 but it's not in this region
            // or we don't have permissions to it.
            if ((ase.statusCode() == HttpStatusCode.MOVED_PERMANENTLY) ||
 "AccessDenied".equals(ase.awsErrorDetails().errorCode())) {
                return true;
            }
            if (ase.statusCode() == HttpStatusCode.NOT_FOUND) {
                return false;
            }
            throw ase;
        }
    }
}
```

- Per i dettagli sull'API, consulta la sezione API Reference. GetBucketAclAWS SDK for Java 2.x

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.

Di seguito vengono mostrati tre esempi di come creare librerie client predefinite URLs e URLs utilizzarle con HTTP:

- Una richiesta HTTP GET che utilizza l'URL con tre librerie client HTTP
- Una richiesta HTTP PUT con metadati nelle intestazioni che utilizza l'URL con tre librerie client HTTP
- Una richiesta HTTP PUT con parametri di query che utilizza l'URL con una libreria client HTTP

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 `HttpURLConnection` (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 con metadati nelle intestazioni 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 `HttpURLConnection` (dalla v1.1) per eseguire il caricamento.

```
   /* Use the JDK HttpURLConnection (since v1.1) class to do the upload. */
   public void useHttpUrlConnectionToPut(String presignedUrlString, File fileToPut,
Map<String, String> metadata) {
```

```
        logger.info("Begin [{}] upload", fileToPut.toString());
        try {
            URL presignedUrl = new URL(presignedUrlString);
            HttpURLConnection connection = (HttpURLConnection)
presignedUrl.openConnection();
            connection.setDoOutput(true);
            metadata.forEach((k, v) -> connection.setRequestProperty("x-amz-meta-" +
k, v));
            connection.setRequestMethod("PUT");
            OutputStream out = connection.getOutputStream();

            try (RandomAccessFile file = new RandomAccessFile(fileToPut, "r");
                 FileChannel inChannel = file.getChannel()) {
                ByteBuffer buffer = ByteBuffer.allocate(8192); //Buffer size is 8k

                while (inChannel.read(buffer) > 0) {
                    buffer.flip();
                    for (int i = 0; i < buffer.limit(); i++) {
                        out.write(buffer.get());
                    }
                    buffer.clear();
                }
            } catch (IOException e) {
                logger.error(e.getMessage(), e);
            }

            out.close();
            connection.getResponseCode();
            logger.info("HTTP response code is " + connection.getResponseCode());

        } catch (S3Exception | IOException e) {
            logger.error(e.getMessage(), e);
        }
    }
```

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);
        }
    }
```

Genera un URL prefirmato con parametri di query per un caricamento, quindi carica un file (richiesta PUT).

Importazioni.

```
import com.example.s3.util.PresignUrlUtils;
import org.slf4j.Logger;
import software.amazon.awssdk.awscore.AwsRequestOverrideConfiguration;
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.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.net.URISyntaxException;
import java.net.URL;
import java.nio.file.Paths;
import java.time.Duration;
import java.util.Map;
import java.util.UUID;
```

Genera l'URL.

```
   /**
    *  Creates a presigned URL to use in a subsequent HTTP PUT request. The code
adds query parameters
    *  to the request instead of using headers. By using query parameters, you do
not need to add the
    *  the parameters as headers when the PUT request is eventually sent.
    *
    * @param bucketName Bucket name where the object will be uploaded.
    * @param keyName Key name of the object that will be uploaded.
    * @param queryParams Query string parameters to be added to the presigned URL.
    * @return
    */
   public String createPresignedUrl(String bucketName, String keyName, Map<String,
String> queryParams) {
       try (S3Presigner presigner = S3Presigner.create()) {
           // Create an override configuration to store the query parameters.
           AwsRequestOverrideConfiguration.Builder overrideConfigurationBuilder =
AwsRequestOverrideConfiguration.builder();

           queryParams.forEach(overrideConfigurationBuilder::putRawQueryParameter);

           PutObjectRequest objectRequest = PutObjectRequest.builder()
                   .bucket(bucketName)
                   .key(keyName)
                   .overrideConfiguration(overrideConfigurationBuilder.build()) //
Add the override configuration.
                   .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();
        }
    }
```

Usa la `SdkHttpClient` classe AWS for Java V2 per eseguire il caricamento.

```
    /**
     * Use the AWS SDK for Java V2 SdkHttpClient class to execute the PUT request.
Since the
     * URL contains the query parameters, no headers are needed for metadata, SSE
settings, or ACL settings.
     *
     * @param presignedUrlString The URL for the PUT request.
     * @param fileToPut File to uplaod
     */
    public void useSdkHttpClientToPut(String presignedUrlString, File fileToPut) {
        logger.info("Begin [{}] upload", fileToPut.toString());

        try {
            URL presignedUrl = new URL(presignedUrlString);

            SdkHttpRequest.Builder requestBuilder = SdkHttpRequest.builder()
                    .method(SdkHttpMethod.PUT)
                    .uri(presignedUrl.toURI());

            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);
        }
    }
```

Creazione di un'applicazione serverless per gestire foto

Nell'esempio di codice seguente viene illustrato come creare un'applicazione serverless che consente agli utenti di gestire le foto mediante 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, consultate 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

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.

```
   /**
    * Aborts all incomplete multipart uploads from the specified S3 bucket.
    * <p>
    * This method retrieves a list of all incomplete multipart uploads in the
specified S3 bucket,
    * and then aborts each of those uploads.
    */
   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

Rilevamento dei DPI nelle immagini

Il seguente esempio di codice mostra come creare un'app che utilizza Amazon Rekognition per rilevare i dispositivi di protezione individuale (DPI) nelle immagini.

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

Rilevamento di oggetti nelle immagini

Il seguente esempio di codice mostra come creare un'app che utilizza Amazon Rekognition per rilevare oggetti per categoria nelle immagini.

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

Rilevamento di persone e oggetti in un video

Il seguente esempio di codice mostra come rilevare persone e oggetti in un video con Amazon Rekognition.

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
- Amazon SNS
- Amazon SQS

Scarica le «directory» di S3

Il seguente esempio di codice mostra come scaricare e filtrare il contenuto delle «directory» del bucket Amazon 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.

Questo esempio mostra come utilizzare S3 TransferManager per AWS SDK for Java 2.x scaricare «directory» da un bucket Amazon S3. Dimostra anche come utilizzarlo nella richiesta. DownloadFilters

```
    /**
     * For standard buckets, S3 provides the illusion of a directory structure
through the use of keys. When you upload
     * an object to an S3 bucket, you specify a key, which is essentially the "path"
to the object. The key can contain
     * forward slashes ("/") to make it appear as if the object is stored in a
directory structure, but this is just a
     * logical representation, not an actual directory.
     * <p><pre>
     * In this example, our S3 bucket contains the following objects:
     *
     * folder1/file1.txt
     * folder1/file2.txt
     * folder1/file3.txt
     * folder2/file1.txt
     * folder2/file2.txt
     * folder2/file3.txt
     * folder3/file1.txt
     * folder3/file2.txt
     * folder3/file3.txt
     *
     * When method `downloadS3Directories` is invoked with
     * `destinationPathURI` set to `/test`, the downloaded
     * directory looks like:
     *
     * |- test
     *    |- folder1
     *       |- file1.txt
     *       |- file2.txt
     *       |- file3.txt
```

```
    *    |- folder3
    *       |- file1.txt
    *       |- file2.txt
    *       |- file3.txt
    * </pre>
    *
    * @param transferManager    An S3TransferManager instance.
    * @param destinationPathURI local directory to hold the downloaded S3
'directories' and files.
    * @param bucketName         The S3 bucket that contains the 'directories' to
download.
    * @return The number of objects (files, in this case) that were downloaded.
    */
   public Integer downloadS3Directories(S3TransferManager transferManager,
                                        URI destinationPathURI, String bucketName)
{

       // Define the filters for which 'directories' we want to download.
       DownloadFilter folder1Filter = (S3Object s3Object) ->
s3Object.key().startsWith("folder1/");
       DownloadFilter folder3Filter = (S3Object s3Object) ->
s3Object.key().startsWith("folder3/");
       DownloadFilter folderFilter = s3Object ->
folder1Filter.or(folder3Filter).test(s3Object);

       DirectoryDownload directoryDownload =
transferManager.downloadDirectory(DownloadDirectoryRequest.builder()
               .destination(Paths.get(destinationPathURI))
               .bucket(bucketName)
               .filter(folderFilter)
               .build());
       CompletedDirectoryDownload completedDirectoryDownload =
directoryDownload.completionFuture().join();

       Integer numFilesInFolder1 =
Paths.get(destinationPathURI).resolve("folder1").toFile().list().length;
       Integer numFilesInFolder3 =
Paths.get(destinationPathURI).resolve("folder3").toFile().list().length;

       try {
           assert numFilesInFolder1 == 3;
           assert numFilesInFolder3 == 3;
```

```
            assert !
Paths.get(destinationPathURI).resolve("folder2").toFile().exists(); // `folder2` was
 not downloaded.
        } catch (AssertionError e) {
            logger.error("An assertion failed.");
        }

        completedDirectoryDownload.failedTransfers()
                .forEach(fail -> logger.warn("Object failed to transfer  [{}]",
 fail.exception().getMessage()));
        return numFilesInFolder1 + numFilesInFolder3;
    }
```

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 DownloadDirectorylla sezione API Reference.AWS SDK for Java 2.x

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 di più 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.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) {
        final String usage = """
            Usage:
                <bucketName> \s

            Where:
                bucketName - The Amazon S3 bucket name.
           """;

        if (args.length != 1) {
```

```
            System.out.println(usage);
            System.exit(1);
        }
        s3LockActions = new S3LockActions();
        bucketName = args[0];
        Scanner scanner = new Scanner(System.in);

        System.out.println(DASHES);
        System.out.println("Welcome to the Amazon Simple Storage Service (S3) Object
Locking Feature 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 rention = DefaultRetention.builder()
           .days(1)
           .mode(ObjectLockRetentionMode.GOVERNANCE)
           .build();

       ObjectLockRule lockRule = ObjectLockRule.builder()
           .defaultRetention(rention)
           .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.objectLockConfiguration().objectLockEnabled());
        System.out.println("\tRule: "+
 response.objectLockConfiguration().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

Gestisci messaggi di grandi dimensioni usando S3

Il seguente esempio di codice mostra come utilizzare la Amazon SQS Extended Client Library per lavorare con messaggi Amazon SQS di grandi dimensioni.

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.amazon.sqs.javamessaging.AmazonSQSExtendedClient;
import com.amazon.sqs.javamessaging.ExtendedClientConfiguration;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.joda.time.DateTime;
import org.joda.time.format.DateTimeFormat;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.BucketLifecycleConfiguration;
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.ExpirationStatus;
import software.amazon.awssdk.services.s3.model.LifecycleExpiration;
import software.amazon.awssdk.services.s3.model.LifecycleRule;
import software.amazon.awssdk.services.s3.model.LifecycleRuleFilter;
import software.amazon.awssdk.services.s3.model.ListObjectVersionsRequest;
import software.amazon.awssdk.services.s3.model.ListObjectVersionsResponse;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Request;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Response;
import
 software.amazon.awssdk.services.s3.model.PutBucketLifecycleConfigurationRequest;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.CreateQueueRequest;
import software.amazon.awssdk.services.sqs.model.CreateQueueResponse;
import software.amazon.awssdk.services.sqs.model.DeleteMessageRequest;
import software.amazon.awssdk.services.sqs.model.DeleteQueueRequest;
import software.amazon.awssdk.services.sqs.model.Message;
import software.amazon.awssdk.services.sqs.model.ReceiveMessageRequest;
import software.amazon.awssdk.services.sqs.model.ReceiveMessageResponse;
import software.amazon.awssdk.services.sqs.model.SendMessageRequest;

import java.util.Arrays;
import java.util.List;
import java.util.UUID;
```

```
/**
 * Example of using Amazon SQS Extended Client Library for Java 2.x.
 */
public class SqsExtendedClientExample {
    private static final Logger logger =
 LoggerFactory.getLogger(SqsExtendedClientExample.class);

    private String s3BucketName;
    private String queueUrl;
    private final String queueName;
    private final S3Client s3Client;
    private final SqsClient sqsExtendedClient;
    private final int messageSize;

    /**
     * Constructor with default clients and message size.
     */
    public SqsExtendedClientExample() {
        this(S3Client.create(), 300000);
    }

    /**
     * Constructor with custom S3 client and message size.
     *
     * @param s3Client The S3 client to use
     * @param messageSize The size of the test message to create
     */
    public SqsExtendedClientExample(S3Client s3Client, int messageSize) {
        this.s3Client = s3Client;
        this.messageSize = messageSize;

        // Generate a unique bucket name.
        this.s3BucketName = UUID.randomUUID() + "-" +
                DateTimeFormat.forPattern("yyMMdd-hhmmss").print(new DateTime());

        // Generate a unique queue name.
        this.queueName = "MyQueue-" + UUID.randomUUID();

        // Configure the SQS extended client.
        final ExtendedClientConfiguration extendedClientConfig = new
 ExtendedClientConfiguration()
                .withPayloadSupportEnabled(s3Client, s3BucketName);
```

```
        this.sqsExtendedClient = new
AmazonSQSExtendedClient(SqsClient.builder().build(), extendedClientConfig);
    }

    public static void main(String[] args) {
        SqsExtendedClientExample example = new SqsExtendedClientExample();
        try {
            example.setup();
            example.sendAndReceiveMessage();
        } finally {
            example.cleanup();
        }
    }

    /**
     * Send a large message and receive it back.
     *
     * @return The received message
     */
    public Message sendAndReceiveMessage() {
        try {
            // Create a large message.
            char[] chars = new char[messageSize];
            Arrays.fill(chars, 'x');
            String largeMessage = new String(chars);

            // Send the message.
            final SendMessageRequest sendMessageRequest =
SendMessageRequest.builder()
                    .queueUrl(queueUrl)
                    .messageBody(largeMessage)
                    .build();

            sqsExtendedClient.sendMessage(sendMessageRequest);
            logger.info("Sent message of size: {}", largeMessage.length());

            // Receive and return the message.
            final ReceiveMessageResponse receiveMessageResponse =
sqsExtendedClient.receiveMessage(
                    ReceiveMessageRequest.builder().queueUrl(queueUrl).build());

            List<Message> messages = receiveMessageResponse.messages();
            if (messages.isEmpty()) {
                throw new RuntimeException("No messages received");
```

```
            }

            Message message = messages.getFirst();
            logger.info("\nMessage received.");
            logger.info("  ID: {}", message.messageId());
            logger.info("  Receipt handle: {}", message.receiptHandle());
            logger.info("  Message body size: {}", message.body().length());
            logger.info("  Message body (first 5 characters): {}",
message.body().substring(0, 5));

            return message;
        } catch (RuntimeException e) {
            logger.error("Error during message processing: {}", e.getMessage(), e);
            throw e;
        }
    }
```

- 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 .
  - CreateBucket
  - PutBucketLifecycleConfiguration
  - ReceiveMessage
  - SendMessage

Analizza URIs

Il seguente esempio di codice mostra come analizzare Amazon URIs S3 per estrarre componenti importanti come il nome del bucket e la chiave dell'oggetto.

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.

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);
        }
    }
```

Elabora le notifiche degli eventi S3

Il seguente esempio di codice mostra come utilizzare le notifiche di eventi S3 in modo orientato agli oggetti.

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.

Questo esempio mostra come elaborare un evento di notifica S3 utilizzando Amazon SQS.

```
    /**
     * This method receives S3 event notifications by using an SqsAsyncClient.
     * After the client receives the messages it deserializes the JSON payload and
logs them. It uses
     * the S3EventNotification class (part of the S3 event notification API for
Java) to deserialize
     * the JSON payload and access the messages in an object-oriented way.
     *
     * @param queueUrl The URL of the AWS SQS queue that receives the S3 event
notifications.
```

```
     * @see <a href="https://sdk.amazonaws.com/java/api/latest/software/amazon/
awssdk/eventnotifications/s3/model/package-summary.html">S3EventNotification API</
a>.
     * <p>
     * To use S3 event notification serialization/deserialization to objects, add
 the following
     * dependency to your Maven pom.xml file.
     * <dependency>
     * <groupId>software.amazon.awssdk</groupId>
     * <artifactId>s3-event-notifications</artifactId>
     * <version><LATEST></version>
     * </dependency>
     * <p>
     * The S3 event notification API became available with version 2.25.11 of the
 Java SDK.
     * <p>
     * This example shows the use of the API with AWS SQS, but it can be used to
 process S3 event notifications
     * in AWS SNS or AWS Lambda as well.
     * <p>
     * Note: The S3EventNotification class does not work with messages routed
 through AWS EventBridge.
     */
    static void processS3Events(String bucketName, String queueUrl, String queueArn)
 {
        try {
            // Configure the bucket to send Object Created and Object Tagging
 notifications to an existing SQS queue.
            s3Client.putBucketNotificationConfiguration(b -> b
                    .notificationConfiguration(ncb -> ncb
                            .queueConfigurations(qcb -> qcb
                                    .events(Event.S3_OBJECT_CREATED,
 Event.S3_OBJECT_TAGGING)
                                    .queueArn(queueArn)))
                            .bucket(bucketName)
            ).join();

            triggerS3EventNotifications(bucketName);
            // Wait for event notifications to propagate.
            Thread.sleep(Duration.ofSeconds(5).toMillis());

            boolean didReceiveMessages = true;
            while (didReceiveMessages) {
                // Display the number of messages that are available in the queue.
```

```
                sqsClient.getQueueAttributes(b -> b
                                .queueUrl(queueUrl)

.attributeNames(QueueAttributeName.APPROXIMATE_NUMBER_OF_MESSAGES)
                        ).thenAccept(attributeResponse ->
                                logger.info("Approximate number of messages in the
queue: {}",

attributeResponse.attributes().get(QueueAttributeName.APPROXIMATE_NUMBER_OF_MESSAGES)))
                        .join();

                // Receive the messages.
                ReceiveMessageResponse response = sqsClient.receiveMessage(b -> b
                        .queueUrl(queueUrl)
                ).get();
                logger.info("Count of received messages: {}",
response.messages().size());
                didReceiveMessages = !response.messages().isEmpty();

                // Create a collection to hold the received message for deletion
                // after we log the messages.
                HashSet<DeleteMessageBatchRequestEntry> messagesToDelete = new
HashSet<>();
                // Process each message.
                response.messages().forEach(message -> {
                    logger.info("Message id: {}", message.messageId());
                    // Deserialize JSON message body to a S3EventNotification object
                    // to access messages in an object-oriented way.
                    S3EventNotification event =
S3EventNotification.fromJson(message.body());

                    // Log the S3 event notification record details.
                    if (event.getRecords() != null) {
                        event.getRecords().forEach(record -> {
                            String eventName = record.getEventName();
                            String key = record.getS3().getObject().getKey();
                            logger.info(record.toString());
                            logger.info("Event name is {} and key is {}", eventName,
key);
                        });
                    }
                    // Add logged messages to collection for batch deletion.
                    messagesToDelete.add(DeleteMessageBatchRequestEntry.builder()
                            .id(message.messageId())
```

```
                        .receiptHandle(message.receiptHandle())
                        .build());
            });
            // Delete messages.
            if (!messagesToDelete.isEmpty()) {
                sqsClient.deleteMessageBatch(DeleteMessageBatchRequest.builder()
                        .queueUrl(queueUrl)
                        .entries(messagesToDelete)
                        .build()
                ).join();
            }
        } // End of while block.
    } catch (InterruptedException | ExecutionException e) {
        throw new RuntimeException(e);
    }
}
```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella Documentazione di riferimento delle API AWS SDK for Java 2.x .
  - DeleteMessageBatch
  - GetQueueAttributes
  - PutBucketNotificationConfiguration
  - ReceiveMessage

Invia notifiche di eventi a EventBridge

Il seguente esempio di codice mostra come abilitare un bucket per inviare notifiche di eventi S3 EventBridge e indirizzarle verso un argomento Amazon SNS e una coda Amazon SQS.

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.

```
   /** This method configures a bucket to send events to AWS EventBridge and
creates a rule
    * to route the S3 object created events to a topic and a queue.
    *
    * @param bucketName Name of existing bucket
    * @param topicArn ARN of existing topic to receive S3 event notifications
    * @param queueArn ARN of existing queue to receive S3 event notifications
    *
    *  An AWS CloudFormation stack sets up the bucket, queue, topic before the
method runs.
    */
   public static String setBucketNotificationToEventBridge(String bucketName,
String topicArn, String queueArn) {
       try {
           // Enable bucket to emit S3 Event notifications to EventBridge.
           s3Client.putBucketNotificationConfiguration(b -> b
                   .bucket(bucketName)
                   .notificationConfiguration(b1 -> b1
                           .eventBridgeConfiguration(
                                   SdkBuilder::build)
                   ).build()).join();

           // Create an EventBridge rule to route Object Created notifications.
           PutRuleRequest putRuleRequest = PutRuleRequest.builder()
                   .name(RULE_NAME)
                   .eventPattern("""
                           {
                             "source": ["aws.s3"],
                             "detail-type": ["Object Created"],
                             "detail": {
                               "bucket": {
                                 "name": ["%s"]
                               }
                             }
                           }
                           """.formatted(bucketName))
                   .build();

           // Add the rule to the default event bus.
           PutRuleResponse putRuleResponse =
eventBridgeClient.putRule(putRuleRequest)
                   .whenComplete((r, t) -> {
                       if (t != null) {
```

```
                            logger.error("Error creating event bus rule: " +
t.getMessage(), t);
                            throw new RuntimeException(t.getCause().getMessage(),
t);
                        }
                        logger.info("Event bus rule creation request sent
successfully. ARN is: {}", r.ruleArn());
                    }).join();

            // Add the existing SNS topic and SQS queue as targets to the rule.
            eventBridgeClient.putTargets(b -> b
                    .eventBusName("default")
                    .rule(RULE_NAME)
                    .targets(List.of (
                            Target.builder()
                                    .arn(queueArn)
                                    .id("Queue")
                                    .build(),
                            Target.builder()
                                    .arn(topicArn)
                                    .id("Topic")
                                    .build())
                            )
                    ).join();
            return putRuleResponse.ruleArn();
        } catch (S3Exception 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 .
  - PutBucketNotificationConfiguration
  - PutRule
  - PutTargets

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 downloadedFileWithPath) {
        DownloadFileRequest downloadFileRequest = DownloadFileRequest.builder()
                .getObjectRequest(b -> b.bucket(bucketName).key(key))
                .addTransferListener(LoggingTransferListener.create())  // Add
listener.
                .destination(Paths.get(downloadedFileWithPath))
                .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();
    }
```

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.

```
    /**
     * Uploads a file to an Amazon S3 bucket using the S3TransferManager.
     *
     * @param filePath the file path of the file to be uploaded
     */
    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();
    }
```

Utilizza l'API S3Client per eseguire un caricamento in più parti.

```
   /**
    * Performs a multipart upload to Amazon S3 using the provided S3 client.
    *
    * @param filePath the path to the file to be uploaded
    */
   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.

```
    /**
     * Uploads a file to an S3 bucket using the S3AsyncClient and enabling multipart
support.
     *
     * @param filePath the local file path of the file to be uploaded
     */
    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.");
    }
```

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 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 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 s33CrtAsyncClient - 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 s33CrtAsyncClient,
 String bucketName, String key) {

        BlockingInputStreamAsyncRequestBody body =
                AsyncRequestBody.forBlockingInputStream(null); // 'null' indicates a
 stream will be provided later.

        CompletableFuture<PutObjectResponse> responseFuture =
                s33CrtAsyncClient.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' CRC32 algoritmo.

```
    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 Amazon S3 Control con SDK for Java 2.x

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando il controllo AWS SDK for Java 2.x con Amazon S3.

Le nozioni di base sono esempi di codice che mostrano come eseguire le operazioni essenziali all'interno di un servizio.

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.

Ogni esempio include un collegamento al codice sorgente completo, dove puoi trovare istruzioni su come configurare ed eseguire il codice nel contesto.

Nozioni di base

Salve Amazon S3 Control

Il seguente esempio di codice mostra come iniziare a usare Amazon S3 Control.

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.auth.credentials.EnvironmentVariableCredentialsProvider;
import software.amazon.awssdk.core.client.config.ClientOverrideConfiguration;
import software.amazon.awssdk.core.retry.RetryMode;
import software.amazon.awssdk.core.retry.RetryPolicy;
import software.amazon.awssdk.http.async.SdkAsyncHttpClient;
import software.amazon.awssdk.http.nio.netty.NettyNioAsyncHttpClient;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3control.S3ControlAsyncClient;
import software.amazon.awssdk.services.s3control.model.JobListDescriptor;
import software.amazon.awssdk.services.s3control.model.JobStatus;
import software.amazon.awssdk.services.s3control.model.ListJobsRequest;
import software.amazon.awssdk.services.s3control.paginators.ListJobsPublisher;
```

```
import java.time.Duration;
import java.util.List;
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.CompletionException;

/**
 * Before running this example:
 * <p/>
 * The SDK must be able to authenticate AWS requests on your behalf. If you have not
 configured
 * authentication for SDKs and tools,see https://docs.aws.amazon.com/sdkref/latest/
guide/access.html in the AWS SDKs and Tools Reference Guide.
 * <p/>
 * You must have a runtime environment configured with the Java SDK.
 * See https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/setup.html in
 the Developer Guide if this is not set up.
 */
public class HelloS3Batch {
    private static S3ControlAsyncClient asyncClient;

    public static void main(String[] args) {
        S3BatchActions actions = new S3BatchActions();
        String accountId = actions.getAccountId();
        try {
            listBatchJobsAsync(accountId)
                .exceptionally(ex -> {
                    System.err.println("List batch jobs failed: " +
 ex.getMessage());
                    return null;
                })
                .join();

        } catch (CompletionException ex) {
            System.err.println("Failed to list batch jobs: " + ex.getMessage());
        }
    }

    /**
     * Retrieves the asynchronous S3 Control client instance.
     * <p>
     * This method creates and returns a singleton instance of the {@link
 S3ControlAsyncClient}. If the instance
     * has not been created yet, it will be initialized with the following
 configuration:
```

```
     * <ul>
     *   <li>Maximum concurrency: 100</li>
     *   <li>Connection timeout: 60 seconds</li>
     *   <li>Read timeout: 60 seconds</li>
     *   <li>Write timeout: 60 seconds</li>
     *   <li>API call timeout: 2 minutes</li>
     *   <li>API call attempt timeout: 90 seconds</li>
     *   <li>Retry policy: 3 retries</li>
     *   <li>Region: US_EAST_1</li>
     *   <li>Credentials provider: {@link EnvironmentVariableCredentialsProvider}</
li>
     * </ul>
     *
     * @return the asynchronous S3 Control client instance
     */
    private static S3ControlAsyncClient getAsyncClient() {
        if (asyncClient == null) {
            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                .maxConcurrency(100)
                .connectionTimeout(Duration.ofSeconds(60))
                .readTimeout(Duration.ofSeconds(60))
                .writeTimeout(Duration.ofSeconds(60))
                .build();

            ClientOverrideConfiguration overrideConfig =
 ClientOverrideConfiguration.builder()
                .apiCallTimeout(Duration.ofMinutes(2))
                .apiCallAttemptTimeout(Duration.ofSeconds(90))
                .retryStrategy(RetryMode.STANDARD)
                .build();

            asyncClient = S3ControlAsyncClient.builder()
                .region(Region.US_EAST_1)
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return asyncClient;
    }

    /**
     * Asynchronously lists batch jobs that have completed for the specified
 account.
     *
```

```
     * @param accountId the ID of the account to list jobs for
     * @return a CompletableFuture that completes when the job listing operation is
finished
     */
    public static CompletableFuture<Void> listBatchJobsAsync(String accountId) {
        ListJobsRequest jobsRequest = ListJobsRequest.builder()
            .jobStatuses(JobStatus.COMPLETE)
            .accountId(accountId)
            .maxResults(10)
            .build();

        ListJobsPublisher publisher =
getAsyncClient().listJobsPaginator(jobsRequest);
        return publisher.subscribe(response -> {
            List<JobListDescriptor> jobs = response.jobs();
            for (JobListDescriptor job : jobs) {
                System.out.println("The job id is " + job.jobId());
                System.out.println("The job priority is " + job.priority());
            }
        }).thenAccept(response -> {
            System.out.println("Listing batch jobs completed");
        }).exceptionally(ex -> {
            System.err.println("Failed to list batch jobs: " + ex.getMessage());
            throw new RuntimeException(ex);
        });
    }
```

- Per i dettagli sull'API, ListJobsconsulta AWS SDK for Java 2.x API Reference.

Argomenti

- Nozioni di base
- Azioni

Nozioni di base

Informazioni di base

Il seguente esempio di codice mostra come apprendere le operazioni di base per Amazon S3 Control.

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.

Scopri le operazioni principali.

```
package com.example.s3.batch;

import software.amazon.awssdk.services.s3.model.S3Exception;
import java.io.IOException;
import java.util.Map;
import java.util.Scanner;
import java.util.UUID;
import java.util.concurrent.CompletionException;

public class S3BatchScenario {

    public static final String DASHES = new String(new char[80]).replace("\0", "-");
    private static final String STACK_NAME = "MyS3Stack";
    public static void main(String[] args) throws IOException {
        S3BatchActions actions = new S3BatchActions();
        String accountId = actions.getAccountId();
        String uuid = java.util.UUID.randomUUID().toString();
        Scanner scanner = new Scanner(System.in);

        System.out.println(DASHES);
        System.out.println("Welcome to the Amazon S3 Batch basics scenario.");
        System.out.println("""
            S3 Batch operations enables efficient and cost-effective processing of
 large-scale
            data stored in Amazon S3. It automatically scales resources to handle
 varying workloads
            without the need for manual intervention.

            One of the key features of S3 Batch is its ability to perform tagging
 operations on objects stored in
            S3 buckets. Users can leverage S3 Batch to apply, update, or remove tags
 on thousands or millions of
```

```
            objects in a single operation, streamlining the management and
 organization of their data.

            This can be particularly useful for tasks such as cost allocation,
 lifecycle management, or
            metadata-driven workflows, where consistent and accurate tagging is
 essential.
            S3 Batch's scalability and serverless nature make it an ideal solution
 for organizations with
            growing data volumes and complex data management requirements.

            This Java program walks you through Amazon S3 Batch operations.

            Let's get started...

            """);
        waitForInputToContinue(scanner);
        // Use CloudFormation to stand up the resource required for this scenario.
        System.out.println("Use CloudFormation to stand up the resource required for
 this scenario.");
        CloudFormationHelper.deployCloudFormationStack(STACK_NAME);

        Map<String, String> stackOutputs =
 CloudFormationHelper.getStackOutputs(STACK_NAME);
        String iamRoleArn = stackOutputs.get("S3BatchRoleArn");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Setup the required bucket for this scenario.");
        waitForInputToContinue(scanner);
        String bucketName = "amzn-s3-demo-bucket-" + UUID.randomUUID(); // Change
 bucket name.
        actions.createBucket(bucketName);
        String reportBucketName = "arn:aws:s3:::"+bucketName;
        String manifestLocation = "arn:aws:s3:::"+bucketName+"/job-manifest.csv";
        System.out.println("Populate the bucket with the required files.");
        String[] fileNames = {"job-manifest.csv", "object-key-1.txt", "object-
key-2.txt", "object-key-3.txt", "object-key-4.txt"};
        actions.uploadFilesToBucket(bucketName, fileNames, actions);
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("1. Create a S3 Batch Job");
```

```
        System.out.println("This job tags all objects listed in the manifest file
with tags");
        waitForInputToContinue(scanner);
        String jobId ;
        try {
            jobId = actions.createS3JobAsync(accountId, iamRoleArn,
manifestLocation, reportBucketName, uuid).join();
            System.out.println("The Job id is " + jobId);

        } catch (S3Exception e) {
            System.err.println("SSM error: " + e.getMessage());
            return;
        } catch (RuntimeException e) {
            System.err.println("Unexpected error: " + e.getMessage());
            return;
        }

        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("2. Update an existing S3 Batch Operations job's
priority");
        System.out.println("""
             In this step, we modify the job priority value. The higher the number,
the higher the priority.
             So, a job with a priority of `30` would have a higher priority than a
job with
             a priority of `20`. This is a common way to represent the priority of a
task
             or job, with higher numbers indicating a higher priority.

             Ensure that the job status allows for priority updates. Jobs in
certain
             states (e.g., Cancelled, Failed, or Completed) cannot have their
priorities
             updated. Only jobs in the Active or Suspended state typically allow
priority
             updates.
             """);

        try {
            actions.updateJobPriorityAsync(jobId, accountId)
                .exceptionally(ex -> {
```

```
                    System.err.println("Update job priority failed: " +
ex.getMessage());
                    return null;
                })
                .join();
        } catch (CompletionException ex) {
            System.err.println("Failed to update job priority: " + ex.getMessage());
        }
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. Cancel the S3 Batch job");
        System.out.print("Do you want to cancel the Batch job? (y/n): ");
        String cancelAns = scanner.nextLine();
        if (cancelAns != null && cancelAns.trim().equalsIgnoreCase("y")) {
            try {
                actions.cancelJobAsync(jobId, accountId)
                    .exceptionally(ex -> {
                        System.err.println("Cancel job failed: " + ex.getMessage());
                        return null;
                    })
                    .join();
            } catch (CompletionException ex) {
                System.err.println("Failed to cancel job: " + ex.getMessage());
            }
        } else {
            System.out.println("Job " +jobId +" was not canceled.");
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. Describe the job that was just created");
        waitForInputToContinue(scanner);
        try {
            actions.describeJobAsync(jobId, accountId)
                .exceptionally(ex -> {
                    System.err.println("Describe job failed: " + ex.getMessage());
                    return null;
                })
                .join();
        } catch (CompletionException ex) {
            System.err.println("Failed to describe job: " + ex.getMessage());
        }
```

```
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. Describe the tags associated with the job");
        waitForInputToContinue(scanner);
        try {
            actions.getJobTagsAsync(jobId, accountId)
                .exceptionally(ex -> {
                    System.err.println("Get job tags failed: " + ex.getMessage());
                    return null;
                })
                .join();
        } catch (CompletionException ex) {
            System.err.println("Failed to get job tags: " + ex.getMessage());
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6. Update Batch Job Tags");
        waitForInputToContinue(scanner);
        try {
            actions.putJobTaggingAsync(jobId, accountId)
                .exceptionally(ex -> {
                    System.err.println("Put job tagging failed: " +
ex.getMessage());
                    return null;
                })
                .join();
        } catch (CompletionException ex) {
            System.err.println("Failed to put job tagging: " + ex.getMessage());
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Delete the Amazon S3 Batch job tagging.");
        System.out.print("Do you want to delete Batch job tagging? (y/n)");
        String delAns = scanner.nextLine();
        if (delAns != null && delAns.trim().equalsIgnoreCase("y")) {
            try {
                actions.deleteBatchJobTagsAsync(jobId, accountId)
                    .exceptionally(ex -> {
                        System.err.println("Delete batch job tags failed: " +
ex.getMessage());
                        return null;
```

```
                    })
                    .join();
            } catch (CompletionException ex) {
                System.err.println("Failed to delete batch job tags: " +
ex.getMessage());
            }
        } else {
            System.out.println("Tagging was not deleted.");
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.print("Do you want to delete the AWS resources used in this
scenario? (y/n)");
        String delResAns = scanner.nextLine();
        if (delResAns != null && delResAns.trim().equalsIgnoreCase("y")) {
            actions.deleteFilesFromBucket(bucketName, fileNames, actions);
            actions.deleteBucketFolderAsync(bucketName);
            actions.deleteBucket(bucketName)
                .thenRun(() -> System.out.println("Bucket deletion completed"))
                .exceptionally(ex -> {
                    System.err.println("Error occurred: " + ex.getMessage());
                    return null;
                });
            CloudFormationHelper.destroyCloudFormationStack(STACK_NAME);
        } else {
            System.out.println("The AWS resources were not deleted.");
        }
        System.out.println("The Amazon S3 Batch scenario has successfully
completed.");
        System.out.println(DASHES);
    }

    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.");
            }
        }
    }


}
```

Una classe d'azione che racchiude le operazioni.

```
public class S3BatchActions {

    private static S3ControlAsyncClient asyncClient;

    private static S3AsyncClient s3AsyncClient ;
    /**
     * Retrieves the asynchronous S3 Control client instance.
     * <p>
     * This method creates and returns a singleton instance of the {@link
 S3ControlAsyncClient}. If the instance
     * has not been created yet, it will be initialized with the following
 configuration:
     * <ul>
     *   <li>Maximum concurrency: 100</li>
     *   <li>Connection timeout: 60 seconds</li>
     *   <li>Read timeout: 60 seconds</li>
     *   <li>Write timeout: 60 seconds</li>
     *   <li>API call timeout: 2 minutes</li>
     *   <li>API call attempt timeout: 90 seconds</li>
     *   <li>Retry policy: 3 retries</li>
     *   <li>Region: US_EAST_1</li>
     *   <li>Credentials provider: {@link EnvironmentVariableCredentialsProvider}</
li>
     * </ul>
     *
     * @return the asynchronous S3 Control client instance
     */
    private static S3ControlAsyncClient getAsyncClient() {
        if (asyncClient == null) {
            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                .maxConcurrency(100)
```

```
                .connectionTimeout(Duration.ofSeconds(60))
                .readTimeout(Duration.ofSeconds(60))
                .writeTimeout(Duration.ofSeconds(60))
                .build();

            ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
                .apiCallTimeout(Duration.ofMinutes(2))
                .apiCallAttemptTimeout(Duration.ofSeconds(90))
                .retryPolicy(RetryPolicy.builder()
                    .numRetries(3)
                    .build())
                .build();

            asyncClient = S3ControlAsyncClient.builder()
                .region(Region.US_EAST_1)
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return asyncClient;
    }

    private static S3AsyncClient getS3AsyncClient() {
        if (asyncClient == null) {
            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                .maxConcurrency(100)
                .connectionTimeout(Duration.ofSeconds(60))
                .readTimeout(Duration.ofSeconds(60))
                .writeTimeout(Duration.ofSeconds(60))
                .build();

            ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
                .apiCallTimeout(Duration.ofMinutes(2))
                .apiCallAttemptTimeout(Duration.ofSeconds(90))
                .retryStrategy(RetryMode.STANDARD)
                .build();

            s3AsyncClient = S3AsyncClient.builder()
                .region(Region.US_EAST_1)
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
                .build();
```

```
        }
        return s3AsyncClient;
    }


    /**
     * Cancels a job asynchronously.
     *
     * @param jobId The ID of the job to be canceled.
     * @param accountId The ID of the account associated with the job.
     * @return A {@link CompletableFuture} that completes when the job status has
been updated to "CANCELLED".
     *         If an error occurs during the update, the returned future will
complete exceptionally.
     */
    public CompletableFuture<Void> cancelJobAsync(String jobId, String accountId) {
        UpdateJobStatusRequest updateJobStatusRequest =
UpdateJobStatusRequest.builder()
            .accountId(accountId)
            .jobId(jobId)
            .requestedJobStatus(String.valueOf(JobStatus.CANCELLED))
            .build();

        return asyncClient.updateJobStatus(updateJobStatusRequest)
            .thenAccept(updateJobStatusResponse -> {
                System.out.println("Job status updated to: " +
updateJobStatusResponse.status());
            })
            .exceptionally(ex -> {
                System.err.println("Failed to cancel job: " + ex.getMessage());
                throw new RuntimeException(ex); // Propagate the exception
            });
    }

    /**
     * Updates the priority of a job asynchronously.
     *
     * @param jobId     the ID of the job to update
     * @param accountId the ID of the account associated with the job
     * @return a {@link CompletableFuture} that represents the asynchronous
operation, which completes when the job priority has been updated or an error has
occurred
     */
```

```
   public CompletableFuture<Void> updateJobPriorityAsync(String jobId, String
accountId) {
       UpdateJobPriorityRequest priorityRequest =
UpdateJobPriorityRequest.builder()
           .accountId(accountId)
           .jobId(jobId)
           .priority(60)
           .build();

       CompletableFuture<Void> future = new CompletableFuture<>();
       getAsyncClient().updateJobPriority(priorityRequest)
           .thenAccept(response -> {
               System.out.println("The job priority was updated");
               future.complete(null); // Complete the CompletableFuture on
successful execution
           })
           .exceptionally(ex -> {
               System.err.println("Failed to update job priority: " +
ex.getMessage());
               future.completeExceptionally(ex); // Complete the CompletableFuture
exceptionally on error
               return null; // Return null to handle the exception
           });

       return future;
   }

   /**
    * Asynchronously retrieves the tags associated with a specific job in an AWS
account.
    *
    * @param jobId     the ID of the job for which to retrieve the tags
    * @param accountId the ID of the AWS account associated with the job
    * @return a {@link CompletableFuture} that completes when the job tags have
been retrieved, or with an exception if the operation fails
    * @throws RuntimeException if an error occurs while retrieving the job tags
    */
   public CompletableFuture<Void> getJobTagsAsync(String jobId, String accountId) {
       GetJobTaggingRequest request = GetJobTaggingRequest.builder()
           .jobId(jobId)
           .accountId(accountId)
           .build();

       return asyncClient.getJobTagging(request)
```

```
            .thenAccept(response -> {
                List<S3Tag> tags = response.tags();
                if (tags.isEmpty()) {
                    System.out.println("No tags found for job ID: " + jobId);
                } else {
                    for (S3Tag tag : tags) {
                        System.out.println("Tag key is: " + tag.key());
                        System.out.println("Tag value is: " + tag.value());
                    }
                }
            })
            .exceptionally(ex -> {
                System.err.println("Failed to get job tags: " + ex.getMessage());
                throw new RuntimeException(ex); // Propagate the exception
            });
    }

    /**
     * Asynchronously deletes the tags associated with a specific batch job.
     *
     * @param jobId     The ID of the batch job whose tags should be deleted.
     * @param accountId The ID of the account associated with the batch job.
     * @return A CompletableFuture that completes when the job tags have been
successfully deleted, or an exception is thrown if the deletion fails.
     */
    public CompletableFuture<Void> deleteBatchJobTagsAsync(String jobId, String
accountId) {
        DeleteJobTaggingRequest jobTaggingRequest =
DeleteJobTaggingRequest.builder()
            .accountId(accountId)
            .jobId(jobId)
            .build();

        return asyncClient.deleteJobTagging(jobTaggingRequest)
            .thenAccept(response -> {
                System.out.println("You have successfully deleted " + jobId + "
tagging.");
            })
            .exceptionally(ex -> {
                System.err.println("Failed to delete job tags: " + ex.getMessage());
                throw new RuntimeException(ex);
            });
    }
```

```
   /**
    * Asynchronously describes the specified job.
    *
    * @param jobId     the ID of the job to describe
    * @param accountId the ID of the AWS account associated with the job
    * @return a {@link CompletableFuture} that completes when the job description
is available
    * @throws RuntimeException if an error occurs while describing the job
    */
   public CompletableFuture<Void> describeJobAsync(String jobId, String accountId)
{
       DescribeJobRequest jobRequest = DescribeJobRequest.builder()
           .jobId(jobId)
           .accountId(accountId)
           .build();

       return getAsyncClient().describeJob(jobRequest)
           .thenAccept(response -> {
               System.out.println("Job ID: " + response.job().jobId());
               System.out.println("Description: " + response.job().description());
               System.out.println("Status: " + response.job().statusAsString());
               System.out.println("Role ARN: " + response.job().roleArn());
               System.out.println("Priority: " + response.job().priority());
               System.out.println("Progress Summary: " +
response.job().progressSummary());

               // Print out details about the job manifest.
               JobManifest manifest = response.job().manifest();
               System.out.println("Manifest Location: " +
manifest.location().objectArn());
               System.out.println("Manifest ETag: " + manifest.location().eTag());

               // Print out details about the job operation.
               JobOperation operation = response.job().operation();
               if (operation.s3PutObjectTagging() != null) {
                   System.out.println("Operation: S3 Put Object Tagging");
                   System.out.println("Tag Set: " +
operation.s3PutObjectTagging().tagSet());
               }

               // Print out details about the job report.
               JobReport report = response.job().report();
               System.out.println("Report Bucket: " + report.bucket());
               System.out.println("Report Prefix: " + report.prefix());
```

```
                System.out.println("Report Format: " + report.format());
                System.out.println("Report Enabled: " + report.enabled());
                System.out.println("Report Scope: " + report.reportScopeAsString());
            })
            .exceptionally(ex -> {
                System.err.println("Failed to describe job: " + ex.getMessage());
                throw new RuntimeException(ex);
            });
    }

    /**
     * Creates an asynchronous S3 job using the AWS Java SDK.
     *
     * @param accountId         the AWS account ID associated with the job
     * @param iamRoleArn        the ARN of the IAM role to be used for the job
     * @param manifestLocation  the location of the job manifest file in S3
     * @param reportBucketName  the name of the S3 bucket to store the job report
     * @param uuid              a unique identifier for the job
     * @return a CompletableFuture that represents the asynchronous creation of the
 S3 job.
     *         The CompletableFuture will return the job ID if the job is created
 successfully,
     *         or throw an exception if there is an error.
     */
    public CompletableFuture<String> createS3JobAsync(String accountId, String
 iamRoleArn,
                                                       String manifestLocation,
 String reportBucketName, String uuid) {

        String[] bucketName = new String[]{""};
        String[] parts = reportBucketName.split(":::");
        if (parts.length > 1) {
            bucketName[0] = parts[1];
        } else {
            System.out.println("The input string does not contain the expected
 format.");
        }

        return CompletableFuture.supplyAsync(() -> getETag(bucketName[0], "job-
manifest.csv"))
            .thenCompose(eTag -> {
                  ArrayList<S3Tag> tagSet = new ArrayList<>();
                S3Tag s3Tag = S3Tag.builder()
                    .key("keyOne")
```

```
                    .value("ValueOne")
                    .build();
                S3Tag s3Tag2 = S3Tag.builder()
                    .key("keyTwo")
                    .value("ValueTwo")
                    .build();
                tagSet.add(s3Tag);
                tagSet.add(s3Tag2);

                S3SetObjectTaggingOperation objectTaggingOperation =
S3SetObjectTaggingOperation.builder()
                    .tagSet(tagSet)
                    .build();

                JobOperation jobOperation = JobOperation.builder()
                    .s3PutObjectTagging(objectTaggingOperation)
                    .build();

                JobManifestLocation jobManifestLocation =
JobManifestLocation.builder()
                    .objectArn(manifestLocation)
                    .eTag(eTag)
                    .build();

                JobManifestSpec manifestSpec = JobManifestSpec.builder()
                    .fieldsWithStrings("Bucket", "Key")
                    .format("S3BatchOperations_CSV_20180820")
                    .build();

                JobManifest jobManifest = JobManifest.builder()
                    .spec(manifestSpec)
                    .location(jobManifestLocation)
                    .build();

                JobReport jobReport = JobReport.builder()
                    .bucket(reportBucketName)
                    .prefix("reports")
                    .format("Report_CSV_20180820")
                    .enabled(true)
                    .reportScope("AllTasks")
                    .build();

                CreateJobRequest jobRequest = CreateJobRequest.builder()
                    .accountId(accountId)
```

```
                    .description("Job created using the AWS Java SDK")
                    .manifest(jobManifest)
                    .operation(jobOperation)
                    .report(jobReport)
                    .priority(42)
                    .roleArn(iamRoleArn)
                    .clientRequestToken(uuid)
                    .confirmationRequired(false)
                    .build();

                // Create the job asynchronously.
                 return getAsyncClient().createJob(jobRequest)
                    .thenApply(CreateJobResponse::jobId);
                 })
                 .handle((jobId, ex) -> {
                    if (ex != null) {
                    Throwable cause = (ex instanceof CompletionException) ?
ex.getCause() : ex;
                    if (cause instanceof S3ControlException) {
                        throw new CompletionException(cause);
                    } else {
                        throw new RuntimeException(cause);
                    }
                }
                return jobId;
            });
    }

    /**
     * Retrieves the ETag (Entity Tag) for an object stored in an Amazon S3 bucket.
     *
     * @param bucketName the name of the Amazon S3 bucket where the object is stored
     * @param key the key (file name) of the object in the Amazon S3 bucket
     * @return the ETag of the object
     */
    public String getETag(String bucketName, String key) {
        S3Client s3Client = S3Client.builder()
            .region(Region.US_EAST_1)
            .build();

        HeadObjectRequest headObjectRequest = HeadObjectRequest.builder()
            .bucket(bucketName)
            .key(key)
            .build();
```

```
        HeadObjectResponse headObjectResponse =
s3Client.headObject(headObjectRequest);
        return headObjectResponse.eTag();
    }

    /**
     * Asynchronously adds tags to a job in the system.
     *
     * @param jobId     the ID of the job to add tags to
     * @param accountId the account ID associated with the job
     * @return a CompletableFuture that completes when the tagging operation is
finished
     */
    public CompletableFuture<Void> putJobTaggingAsync(String jobId, String
accountId) {
        S3Tag departmentTag = S3Tag.builder()
            .key("department")
            .value("Marketing")
            .build();

        S3Tag fiscalYearTag = S3Tag.builder()
            .key("FiscalYear")
            .value("2020")
            .build();

        PutJobTaggingRequest putJobTaggingRequest = PutJobTaggingRequest.builder()
            .jobId(jobId)
            .accountId(accountId)
            .tags(departmentTag, fiscalYearTag)
            .build();

        return asyncClient.putJobTagging(putJobTaggingRequest)
            .thenRun(() -> {
                System.out.println("Additional Tags were added to job " + jobId);
            })
            .exceptionally(ex -> {
                System.err.println("Failed to add tags to job: " + ex.getMessage());
                throw new RuntimeException(ex); // Propagate the exception
            });
    }

    // Setup the S3 bucket required for this scenario.
    /**
```

```
     * Creates an Amazon S3 bucket with the specified name.
     *
     * @param bucketName the name of the S3 bucket to create
     * @throws S3Exception if there is an error creating the bucket
     */
    public void createBucket(String bucketName) {
        try {
            S3Client s3Client = S3Client.builder()
                .region(Region.US_EAST_1)
                .build();


            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);
        }
    }

    /**
     * Uploads a file to an Amazon S3 bucket asynchronously.
     *
     * @param bucketName the name of the S3 bucket to upload the file to
     * @param fileName the name of the file to be uploaded
     * @throws RuntimeException if an error occurs during the file upload
     */
    public void populateBucket(String bucketName, String fileName) {
        // Define the path to the directory.
```

```
        Path filePath = Paths.get("src/main/resources/batch/",
 fileName).toAbsolutePath();
        PutObjectRequest putOb = PutObjectRequest.builder()
            .bucket(bucketName)
            .key(fileName)
            .build();

        CompletableFuture<PutObjectResponse> future =
 getS3AsyncClient().putObject(putOb, AsyncRequestBody.fromFile(filePath));
        future.whenComplete((result, ex) -> {
            if (ex != null) {
                System.err.println("Error uploading file: " + ex.getMessage());
            } else {
                System.out.println("Successfully placed " + fileName + " into bucket
 " + bucketName);
            }
        }).join();
    }


    // Update the bucketName in CSV.
    public void updateCSV(String newValue) {
        Path csvFilePath = Paths.get("src/main/resources/batch/job-
manifest.csv").toAbsolutePath();
        try {
            // Read all lines from the CSV file.
            List<String> lines = Files.readAllLines(csvFilePath);

            // Update the first value in each line.
            List<String> updatedLines = lines.stream()
                .map(line -> {
                    String[] parts = line.split(",");
                    parts[0] = newValue;
                    return String.join(",", parts);
                })
                .collect(Collectors.toList());

            // Write the updated lines back to the CSV file
            Files.write(csvFilePath, updatedLines);
            System.out.println("CSV file updated successfully.");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
```

```
   /**
    * Deletes an object from an Amazon S3 bucket asynchronously.
    *
    * @param bucketName The name of the S3 bucket where the object is stored.
    * @param objectName The name of the object to be deleted.
    * @return A {@link CompletableFuture} that completes when the object has been
deleted,
    *         or throws a {@link RuntimeException} if an error occurs during the
deletion.
    */
   public CompletableFuture<Void> deleteBucketObjects(String bucketName, String
objectName) {
       ArrayList<ObjectIdentifier> toDelete = new ArrayList<>();
       toDelete.add(ObjectIdentifier.builder()
           .key(objectName)
           .build());

       DeleteObjectsRequest dor = DeleteObjectsRequest.builder()
           .bucket(bucketName)
           .delete(Delete.builder()
               .objects(toDelete).build())
           .build();

       return getS3AsyncClient().deleteObjects(dor)
           .thenAccept(result -> {
               System.out.println("The object was deleted!");
           })
           .exceptionally(ex -> {
               throw new RuntimeException("Error deleting object: " +
ex.getMessage(), ex);
           });
   }

   /**
    * Deletes a folder and all its contents asynchronously from an Amazon S3
bucket.
    *
    * @param bucketName the name of the S3 bucket containing the folder to be
deleted
    * @return a {@link CompletableFuture} that completes when the folder and its
contents have been deleted
    * @throws RuntimeException if any error occurs during the deletion process
    */
```

```
   public void deleteBucketFolderAsync(String bucketName) {
       String folderName = "reports/";
       ListObjectsV2Request request = ListObjectsV2Request.builder()
           .bucket(bucketName)
           .prefix(folderName)
           .build();

       CompletableFuture<ListObjectsV2Response> listObjectsFuture =
getS3AsyncClient().listObjectsV2(request);
       listObjectsFuture.thenCompose(response -> {
           List<CompletableFuture<DeleteObjectResponse>> deleteFutures =
response.contents().stream()
               .map(obj -> {
                   DeleteObjectRequest deleteRequest =
DeleteObjectRequest.builder()
                       .bucket(bucketName)
                       .key(obj.key())
                       .build();
                   return getS3AsyncClient().deleteObject(deleteRequest)
                       .thenApply(deleteResponse -> {
                           System.out.println("Deleted object: " + obj.key());
                           return deleteResponse;
                       });
               })
               .collect(Collectors.toList());

           return CompletableFuture.allOf(deleteFutures.toArray(new
CompletableFuture[0]))
               .thenCompose(v -> {
                   // Delete the folder.
                   DeleteObjectRequest deleteRequest =
DeleteObjectRequest.builder()
                       .bucket(bucketName)
                       .key(folderName)
                       .build();
                   return getS3AsyncClient().deleteObject(deleteRequest)
                       .thenApply(deleteResponse -> {
                           System.out.println("Deleted folder: " + folderName);
                           return deleteResponse;
                       });
               });
       }).join();
   }
```

```
    /**
     * Deletes an Amazon S3 bucket.
     *
     * @param bucketName the name of the bucket to delete
     * @return a {@link CompletableFuture} that completes when the bucket has been
deleted, or exceptionally if there is an error
     * @throws RuntimeException if there is an error deleting the bucket
     */
    public CompletableFuture<Void> deleteBucket(String bucketName) {
        S3AsyncClient s3Client = getS3AsyncClient();
        return s3Client.deleteBucket(DeleteBucketRequest.builder()
                .bucket(bucketName)
                .build())
            .thenAccept(deleteBucketResponse -> {
                System.out.println(bucketName + " was deleted");
            })
            .exceptionally(ex -> {
                // Handle the exception or rethrow it.
                throw new RuntimeException("Failed to delete bucket: " + bucketName,
ex);
            });
    }

    /**
     * Uploads a set of files to an Amazon S3 bucket.
     *
     * @param bucketName the name of the S3 bucket to upload the files to
     * @param fileNames an array of file names to be uploaded
     * @param actions an instance of {@link S3BatchActions} that provides the
implementation for the necessary S3 operations
     * @throws IOException if there's an error creating the text files or uploading
the files to the S3 bucket
     */
    public static void uploadFilesToBucket(String bucketName, String[] fileNames,
S3BatchActions actions) throws IOException {
        actions.updateCSV(bucketName);
        createTextFiles(fileNames);
        for (String fileName : fileNames) {
            actions.populateBucket(bucketName, fileName);
        }
        System.out.println("All files are placed in the S3 bucket " + bucketName);
    }

    /**
```

```
    * Deletes the specified files from the given S3 bucket.
    *
    * @param bucketName the name of the S3 bucket
    * @param fileNames an array of file names to be deleted from the bucket
    * @param actions the S3BatchActions instance to be used for the file deletion
    * @throws IOException if an I/O error occurs during the file deletion
    */
   public void deleteFilesFromBucket(String bucketName, String[] fileNames,
S3BatchActions actions) throws IOException {
       for (String fileName : fileNames) {
                  actions.deleteBucketObjects(bucketName, fileName)
                 .thenRun(() -> System.out.println("Object deletion completed"))
                 .exceptionally(ex -> {
                     System.err.println("Error occurred: " + ex.getMessage());
                     return null;
                 });
       }
       System.out.println("All files have been deleted from the bucket " +
bucketName);
   }

   public static void createTextFiles(String[] fileNames) {
       String currentDirectory = System.getProperty("user.dir");
       String directoryPath = currentDirectory + "\\src\\main\\resources\\batch";
       Path path = Paths.get(directoryPath);

       try {
           // Create the directory if it doesn't exist.
           if (Files.notExists(path)) {
               Files.createDirectories(path);
               System.out.println("Created directory: " + path.toString());
           } else {
               System.out.println("Directory already exists: " + path.toString());
           }

           for (String fileName : fileNames) {
               // Check if the file is a .txt file.
               if (fileName.endsWith(".txt")) {
                   // Define the path for the new file.
                   Path filePath = path.resolve(fileName);
                   System.out.println("Attempting to create file: " +
filePath.toString());

                   // Create and write content to the new file.
```

```
                    Files.write(filePath, "This is a test".getBytes());

                    // Verify the file was created.
                    if (Files.exists(filePath)) {
                        System.out.println("Successfully created file: " +
 filePath.toString());
                    } else {
                        System.out.println("Failed to create file: " +
 filePath.toString());
                    }
                }
            }

        } catch (IOException e) {
            System.err.println("An error occurred: " + e.getMessage());
            e.printStackTrace();
        }
    }

    public String getAccountId() {
        StsClient stsClient = StsClient.builder()
            .region(Region.US_EAST_1)
            .build();

        GetCallerIdentityResponse callerIdentityResponse =
 stsClient.getCallerIdentity();
        return callerIdentityResponse.account();
    }
}
```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella Documentazione di riferimento delle API AWS SDK for Java 2.x .
  - CreateJob
  - DeleteJobTagging
  - DescribeJob
  - GetJobTagging
  - ListJobs
  - PutJobTagging
  - UpdateJobPriority

- UpdateJobStatus

Azioni

`CreateJob`

Il seguente esempio di codice mostra come usare. `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.

Crea un lavoro S3 asincrono.

```
   /**
    * Creates an asynchronous S3 job using the AWS Java SDK.
    *
    * @param accountId         the AWS account ID associated with the job
    * @param iamRoleArn        the ARN of the IAM role to be used for the job
    * @param manifestLocation  the location of the job manifest file in S3
    * @param reportBucketName  the name of the S3 bucket to store the job report
    * @param uuid              a unique identifier for the job
    * @return a CompletableFuture that represents the asynchronous creation of the
S3 job.
    *         The CompletableFuture will return the job ID if the job is created
successfully,
    *         or throw an exception if there is an error.
    */
   public CompletableFuture<String> createS3JobAsync(String accountId, String
iamRoleArn,
                                                      String manifestLocation,
String reportBucketName, String uuid) {

       String[] bucketName = new String[]{""};
       String[] parts = reportBucketName.split(":::");
       if (parts.length > 1) {
           bucketName[0] = parts[1];
       } else {
```

```
            System.out.println("The input string does not contain the expected
 format.");
        }

        return CompletableFuture.supplyAsync(() -> getETag(bucketName[0], "job-
manifest.csv"))
            .thenCompose(eTag -> {
                  ArrayList<S3Tag> tagSet = new ArrayList<>();
                S3Tag s3Tag = S3Tag.builder()
                    .key("keyOne")
                    .value("ValueOne")
                    .build();
                S3Tag s3Tag2 = S3Tag.builder()
                    .key("keyTwo")
                    .value("ValueTwo")
                    .build();
                tagSet.add(s3Tag);
                tagSet.add(s3Tag2);

                S3SetObjectTaggingOperation objectTaggingOperation =
 S3SetObjectTaggingOperation.builder()
                    .tagSet(tagSet)
                    .build();

                JobOperation jobOperation = JobOperation.builder()
                    .s3PutObjectTagging(objectTaggingOperation)
                    .build();

                JobManifestLocation jobManifestLocation =
 JobManifestLocation.builder()
                    .objectArn(manifestLocation)
                    .eTag(eTag)
                    .build();

                JobManifestSpec manifestSpec = JobManifestSpec.builder()
                    .fieldsWithStrings("Bucket", "Key")
                    .format("S3BatchOperations_CSV_20180820")
                    .build();

                JobManifest jobManifest = JobManifest.builder()
                    .spec(manifestSpec)
                    .location(jobManifestLocation)
                    .build();
```

```
                JobReport jobReport = JobReport.builder()
                    .bucket(reportBucketName)
                    .prefix("reports")
                    .format("Report_CSV_20180820")
                    .enabled(true)
                    .reportScope("AllTasks")
                    .build();

                CreateJobRequest jobRequest = CreateJobRequest.builder()
                    .accountId(accountId)
                    .description("Job created using the AWS Java SDK")
                    .manifest(jobManifest)
                    .operation(jobOperation)
                    .report(jobReport)
                    .priority(42)
                    .roleArn(iamRoleArn)
                    .clientRequestToken(uuid)
                    .confirmationRequired(false)
                    .build();

                // Create the job asynchronously.
                 return getAsyncClient().createJob(jobRequest)
                    .thenApply(CreateJobResponse::jobId);
                 })
                 .handle((jobId, ex) -> {
                    if (ex != null) {
                    Throwable cause = (ex instanceof CompletionException) ?
ex.getCause() : ex;
                    if (cause instanceof S3ControlException) {
                        throw new CompletionException(cause);
                    } else {
                        throw new RuntimeException(cause);
                    }
                }
                return jobId;
            });
    }
```

Crea un processo di mantenimento della conformità.

```
    /**
```

```
     * Creates a compliance retention job in Amazon S3 Control.
     * <p>
     * A compliance retention job in Amazon S3 Control is a feature that allows you
 to
     * set a retention period for objects stored in an S3 bucket.
     * This feature is particularly useful for organizations that need to comply
 with
     * regulatory requirements or internal policies that mandate the retention of
 data for
     * a specific duration.
     *
     * @param s3ControlClient The S3ControlClient instance to use for the API call.
     * @return The job ID of the created compliance retention job.
     */
    public static String createComplianceRetentionJob(final S3ControlClient
 s3ControlClient, String roleArn, String bucketName, String accountId) {
        final String manifestObjectArn = "arn:aws:s3:::amzn-s3-demo-manifest-bucket/
compliance-objects-manifest.csv";
        final String manifestObjectVersionId = "your-object-version-Id";

        Instant jan2025 = Instant.parse("2025-01-01T00:00:00Z");
        JobOperation jobOperation = JobOperation.builder()
            .s3PutObjectRetention(S3SetObjectRetentionOperation.builder()
                .retention(S3Retention.builder()
                    .mode(S3ObjectLockRetentionMode.COMPLIANCE)
                    .retainUntilDate(jan2025)
                    .build())
                .build())
            .build();

        JobManifestLocation manifestLocation = JobManifestLocation.builder()
            .objectArn(manifestObjectArn)
            .eTag(manifestObjectVersionId)
            .build();

        JobManifestSpec manifestSpec = JobManifestSpec.builder()
            .fieldsWithStrings("Bucket", "Key")
            .format("S3BatchOperations_CSV_20180820")
            .build();

        JobManifest manifestToPublicApi = JobManifest.builder()
            .location(manifestLocation)
            .spec(manifestSpec)
            .build();
```

```
        // Report details.
        final String jobReportBucketArn = "arn:aws:s3:::" + bucketName;
        final String jobReportPrefix = "reports/compliance-objects-bops";

        JobReport jobReport = JobReport.builder()
            .enabled(true)
            .reportScope(JobReportScope.ALL_TASKS)
            .bucket(jobReportBucketArn)
            .prefix(jobReportPrefix)
            .format(JobReportFormat.REPORT_CSV_20180820)
            .build();

        final Boolean requiresConfirmation = true;
        final int priority = 10;
        CreateJobRequest request = CreateJobRequest.builder()
            .accountId(accountId)
            .description("Set compliance retain-until to 1 Jan 2025")
            .manifest(manifestToPublicApi)
            .operation(jobOperation)
            .priority(priority)
            .roleArn(roleArn)
            .report(jobReport)
            .confirmationRequired(requiresConfirmation)
            .build();

        // Create the job and get the result.
        CreateJobResponse result = s3ControlClient.createJob(request);
        return result.jobId();
    }
```

Crea un posto di lavoro sospeso per motivi legali.

```
    /**
     * Creates a compliance retention job in Amazon S3 Control.
     * <p>
     * A compliance retention job in Amazon S3 Control is a feature that allows you
to
     * set a retention period for objects stored in an S3 bucket.
     * This feature is particularly useful for organizations that need to comply
with
```

```
     * regulatory requirements or internal policies that mandate the retention of
 data for
     * a specific duration.
     *
     * @param s3ControlClient The S3ControlClient instance to use for the API call.
     * @return The job ID of the created compliance retention job.
     */
    public static String createComplianceRetentionJob(final S3ControlClient
 s3ControlClient, String roleArn, String bucketName, String accountId) {
        final String manifestObjectArn = "arn:aws:s3:::amzn-s3-demo-manifest-bucket/
compliance-objects-manifest.csv";
        final String manifestObjectVersionId = "your-object-version-Id";

        Instant jan2025 = Instant.parse("2025-01-01T00:00:00Z");
        JobOperation jobOperation = JobOperation.builder()
            .s3PutObjectRetention(S3SetObjectRetentionOperation.builder()
                .retention(S3Retention.builder()
                    .mode(S3ObjectLockRetentionMode.COMPLIANCE)
                    .retainUntilDate(jan2025)
                    .build())
                .build())
            .build();

        JobManifestLocation manifestLocation = JobManifestLocation.builder()
            .objectArn(manifestObjectArn)
            .eTag(manifestObjectVersionId)
            .build();

        JobManifestSpec manifestSpec = JobManifestSpec.builder()
            .fieldsWithStrings("Bucket", "Key")
            .format("S3BatchOperations_CSV_20180820")
            .build();

        JobManifest manifestToPublicApi = JobManifest.builder()
            .location(manifestLocation)
            .spec(manifestSpec)
            .build();

        // Report details.
        final String jobReportBucketArn = "arn:aws:s3:::" + bucketName;
        final String jobReportPrefix = "reports/compliance-objects-bops";

        JobReport jobReport = JobReport.builder()
            .enabled(true)
```

```
            .reportScope(JobReportScope.ALL_TASKS)
            .bucket(jobReportBucketArn)
            .prefix(jobReportPrefix)
            .format(JobReportFormat.REPORT_CSV_20180820)
            .build();

        final Boolean requiresConfirmation = true;
        final int priority = 10;
        CreateJobRequest request = CreateJobRequest.builder()
            .accountId(accountId)
            .description("Set compliance retain-until to 1 Jan 2025")
            .manifest(manifestToPublicApi)
            .operation(jobOperation)
            .priority(priority)
            .roleArn(roleArn)
            .report(jobReport)
            .confirmationRequired(requiresConfirmation)
            .build();

        // Create the job and get the result.
        CreateJobResponse result = s3ControlClient.createJob(request);
        return result.jobId();
    }
```

Crea un nuovo lavoro di mantenimento della governance.

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateGovernanceRetentionJob {

    public static void main(String[]args) throws ParseException {
        final String usage = """

            Usage:
                <manifestObjectArn> <jobReportBucketArn> <roleArn> <accountId>
 <manifestObjectVersionId>
```

```
            Where:
                manifestObjectArn - The Amazon Resource Name (ARN) of the S3 object
that contains the manifest file for the governance objects.\s
                bucketName - The ARN of the S3 bucket where the job report will be
stored.
                roleArn - The ARN of the IAM role that will be used to perform the
governance retention operation.
                accountId - Your AWS account Id.
                manifestObjectVersionId =  A unique value that is used as the `eTag`
property of the `JobManifestLocation` object.
            """;

        if (args.length != 4) {
            System.out.println(usage);
            return;
        }

        String manifestObjectArn = args[0];
        String jobReportBucketArn = args[1];
        String roleArn = args[2];
        String accountId = args[3];
        String manifestObjectVersionId = args[4];

        S3ControlClient s3ControlClient = S3ControlClient.create();
        createGovernanceRetentionJob(s3ControlClient, manifestObjectArn,
jobReportBucketArn, roleArn, accountId, manifestObjectVersionId);
    }

    public static String createGovernanceRetentionJob(final S3ControlClient
s3ControlClient, String manifestObjectArn, String jobReportBucketArn, String
roleArn, String accountId, String manifestObjectVersionId) throws ParseException {
        final JobManifestLocation manifestLocation = JobManifestLocation.builder()
            .objectArn(manifestObjectArn)
            .eTag(manifestObjectVersionId)
            .build();

        final JobManifestSpec manifestSpec = JobManifestSpec.builder()
            .format(JobManifestFormat.S3_BATCH_OPERATIONS_CSV_20180820)
            .fields(Arrays.asList(JobManifestFieldName.BUCKET,
JobManifestFieldName.KEY))
            .build();

        final JobManifest manifestToPublicApi = JobManifest.builder()
```

```
            .location(manifestLocation)
            .spec(manifestSpec)
            .build();

        final String jobReportPrefix = "reports/governance-objects";
        final JobReport jobReport = JobReport.builder()
            .enabled(true)
            .reportScope(JobReportScope.ALL_TASKS)
            .bucket(jobReportBucketArn)
            .prefix(jobReportPrefix)
            .format(JobReportFormat.REPORT_CSV_20180820)
            .build();

        final SimpleDateFormat format = new SimpleDateFormat("dd/MM/yyyy");
        final Date jan30th = format.parse("30/01/2025");

        final S3SetObjectRetentionOperation s3SetObjectRetentionOperation =
S3SetObjectRetentionOperation.builder()
            .retention(S3Retention.builder()
                .mode(S3ObjectLockRetentionMode.GOVERNANCE)
                .retainUntilDate(jan30th.toInstant())
                .build())
            .build();

        final JobOperation jobOperation = JobOperation.builder()
            .s3PutObjectRetention(s3SetObjectRetentionOperation)
            .build();

        final Boolean requiresConfirmation = true;
        final int priority = 10;

        final CreateJobRequest request = CreateJobRequest.builder()
            .accountId(accountId)
            .description("Put governance retention")
            .manifest(manifestToPublicApi)
            .operation(jobOperation)
            .priority(priority)
            .roleArn(roleArn)
            .report(jobReport)
            .confirmationRequired(requiresConfirmation)
            .build();

        final CreateJobResponse result = s3ControlClient.createJob(request);
        return result.jobId();
```

```
    }
}
```

- Per i dettagli sull'API, consulta la CreateJobsezione AWS SDK for Java 2.x API Reference.

`DeleteJobTagging`

Il seguente esempio di codice mostra come utilizzare`DeleteJobTagging`.

SDK per Java 2.x

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
   /**
    * Asynchronously deletes the tags associated with a specific batch job.
    *
    * @param jobId     The ID of the batch job whose tags should be deleted.
    * @param accountId The ID of the account associated with the batch job.
    * @return A CompletableFuture that completes when the job tags have been
successfully deleted, or an exception is thrown if the deletion fails.
    */
   public CompletableFuture<Void> deleteBatchJobTagsAsync(String jobId, String
accountId) {
       DeleteJobTaggingRequest jobTaggingRequest =
DeleteJobTaggingRequest.builder()
           .accountId(accountId)
           .jobId(jobId)
           .build();

       return asyncClient.deleteJobTagging(jobTaggingRequest)
           .thenAccept(response -> {
               System.out.println("You have successfully deleted " + jobId + "
tagging.");
           })
           .exceptionally(ex -> {
               System.err.println("Failed to delete job tags: " + ex.getMessage());
```

```
                throw new RuntimeException(ex);
            });
    }
```

- Per i dettagli sull'API, DeleteJobTaggingconsulta AWS SDK for Java 2.x API Reference.

`DescribeJob`

Il seguente esempio di codice mostra come utilizzare`DescribeJob`.

SDK per Java 2.x

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
    /**
     * Asynchronously describes the specified job.
     *
     * @param jobId     the ID of the job to describe
     * @param accountId the ID of the AWS account associated with the job
     * @return a {@link CompletableFuture} that completes when the job description
is available
     * @throws RuntimeException if an error occurs while describing the job
     */
    public CompletableFuture<Void> describeJobAsync(String jobId, String accountId)
{
        DescribeJobRequest jobRequest = DescribeJobRequest.builder()
            .jobId(jobId)
            .accountId(accountId)
            .build();

        return getAsyncClient().describeJob(jobRequest)
            .thenAccept(response -> {
                System.out.println("Job ID: " + response.job().jobId());
                System.out.println("Description: " + response.job().description());
                System.out.println("Status: " + response.job().statusAsString());
                System.out.println("Role ARN: " + response.job().roleArn());
```

```
                System.out.println("Priority: " + response.job().priority());
                System.out.println("Progress Summary: " +
response.job().progressSummary());

                // Print out details about the job manifest.
                JobManifest manifest = response.job().manifest();
                System.out.println("Manifest Location: " +
manifest.location().objectArn());
                System.out.println("Manifest ETag: " + manifest.location().eTag());

                // Print out details about the job operation.
                JobOperation operation = response.job().operation();
                if (operation.s3PutObjectTagging() != null) {
                    System.out.println("Operation: S3 Put Object Tagging");
                    System.out.println("Tag Set: " +
operation.s3PutObjectTagging().tagSet());
                }

                // Print out details about the job report.
                JobReport report = response.job().report();
                System.out.println("Report Bucket: " + report.bucket());
                System.out.println("Report Prefix: " + report.prefix());
                System.out.println("Report Format: " + report.format());
                System.out.println("Report Enabled: " + report.enabled());
                System.out.println("Report Scope: " + report.reportScopeAsString());
            })
            .exceptionally(ex -> {
                System.err.println("Failed to describe job: " + ex.getMessage());
                throw new RuntimeException(ex);
            });
    }
```

- Per i dettagli sull'API, DescribeJobconsulta AWS SDK for Java 2.x API Reference.

`GetJobTagging`

Il seguente esempio di codice mostra come utilizzare`GetJobTagging`.

SDK per Java 2.x

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
   /**
    * Asynchronously retrieves the tags associated with a specific job in an AWS
account.
    *
    * @param jobId     the ID of the job for which to retrieve the tags
    * @param accountId the ID of the AWS account associated with the job
    * @return a {@link CompletableFuture} that completes when the job tags have
been retrieved, or with an exception if the operation fails
    * @throws RuntimeException if an error occurs while retrieving the job tags
    */
   public CompletableFuture<Void> getJobTagsAsync(String jobId, String accountId) {
       GetJobTaggingRequest request = GetJobTaggingRequest.builder()
           .jobId(jobId)
           .accountId(accountId)
           .build();

       return asyncClient.getJobTagging(request)
           .thenAccept(response -> {
               List<S3Tag> tags = response.tags();
               if (tags.isEmpty()) {
                   System.out.println("No tags found for job ID: " + jobId);
               } else {
                   for (S3Tag tag : tags) {
                       System.out.println("Tag key is: " + tag.key());
                       System.out.println("Tag value is: " + tag.value());
                   }
               }
           })
           .exceptionally(ex -> {
               System.err.println("Failed to get job tags: " + ex.getMessage());
               throw new RuntimeException(ex); // Propagate the exception
           });
   }
```

- Per i dettagli sull'API, GetJobTaggingconsulta AWS SDK for Java 2.x API Reference.

`PutJobTagging`

Il seguente esempio di codice mostra come utilizzare`PutJobTagging`.

SDK per Java 2.x

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
   /**
    * Asynchronously adds tags to a job in the system.
    *
    * @param jobId     the ID of the job to add tags to
    * @param accountId the account ID associated with the job
    * @return a CompletableFuture that completes when the tagging operation is
finished
    */
   public CompletableFuture<Void> putJobTaggingAsync(String jobId, String
accountId) {
       S3Tag departmentTag = S3Tag.builder()
           .key("department")
           .value("Marketing")
           .build();

       S3Tag fiscalYearTag = S3Tag.builder()
           .key("FiscalYear")
           .value("2020")
           .build();

       PutJobTaggingRequest putJobTaggingRequest = PutJobTaggingRequest.builder()
           .jobId(jobId)
           .accountId(accountId)
           .tags(departmentTag, fiscalYearTag)
           .build();

       return asyncClient.putJobTagging(putJobTaggingRequest)
           .thenRun(() -> {
```

```
                System.out.println("Additional Tags were added to job " + jobId);
            })
            .exceptionally(ex -> {
                System.err.println("Failed to add tags to job: " + ex.getMessage());
                throw new RuntimeException(ex); // Propagate the exception
            });
    }
```

- Per i dettagli sull'API, PutJobTaggingconsulta AWS SDK for Java 2.x API Reference.

`UpdateJobPriority`

Il seguente esempio di codice mostra come utilizzare`UpdateJobPriority`.

SDK per Java 2.x

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
    /**
     * Updates the priority of a job asynchronously.
     *
     * @param jobId     the ID of the job to update
     * @param accountId the ID of the account associated with the job
     * @return a {@link CompletableFuture} that represents the asynchronous
operation, which completes when the job priority has been updated or an error has
occurred
     */
    public CompletableFuture<Void> updateJobPriorityAsync(String jobId, String
accountId) {
        UpdateJobPriorityRequest priorityRequest =
UpdateJobPriorityRequest.builder()
            .accountId(accountId)
            .jobId(jobId)
            .priority(60)
            .build();

        CompletableFuture<Void> future = new CompletableFuture<>();
```

```
        getAsyncClient().updateJobPriority(priorityRequest)
            .thenAccept(response -> {
                System.out.println("The job priority was updated");
                future.complete(null); // Complete the CompletableFuture on
successful execution
            })
            .exceptionally(ex -> {
                System.err.println("Failed to update job priority: " +
ex.getMessage());
                future.completeExceptionally(ex); // Complete the CompletableFuture
exceptionally on error
                return null; // Return null to handle the exception
            });

        return future;
    }
```

- Per i dettagli sull'API, UpdateJobPriorityconsulta AWS SDK for Java 2.x API Reference.

`UpdateJobStatus`

Il seguente esempio di codice mostra come utilizzare`UpdateJobStatus`.

SDK per Java 2.x

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
    /**
     * Cancels a job asynchronously.
     *
     * @param jobId The ID of the job to be canceled.
     * @param accountId The ID of the account associated with the job.
     * @return A {@link CompletableFuture} that completes when the job status has
been updated to "CANCELLED".
     *         If an error occurs during the update, the returned future will
complete exceptionally.
     */
```

```
    public CompletableFuture<Void> cancelJobAsync(String jobId, String accountId) {
        UpdateJobStatusRequest updateJobStatusRequest =
UpdateJobStatusRequest.builder()
            .accountId(accountId)
            .jobId(jobId)
            .requestedJobStatus(String.valueOf(JobStatus.CANCELLED))
            .build();

        return asyncClient.updateJobStatus(updateJobStatusRequest)
            .thenAccept(updateJobStatusResponse -> {
                System.out.println("Job status updated to: " +
updateJobStatusResponse.status());
            })
            .exceptionally(ex -> {
                System.err.println("Failed to cancel job: " + ex.getMessage());
                throw new RuntimeException(ex); // Propagate the exception
            });
    }
```

- Per i dettagli sull'API, UpdateJobStatusconsulta AWS SDK for Java 2.x API Reference.

Esempi di S3 Directory Bucket 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 S3 Directory Buckets.

Le nozioni di base sono esempi di codice che mostrano come eseguire le operazioni essenziali all'interno di un servizio.

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.

Gli scenari sono esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio o combinate con altri Servizi AWS.

Ogni esempio include un collegamento al codice sorgente completo, dove puoi trovare istruzioni su come configurare ed eseguire il codice nel contesto.

Nozioni di base

Ciao, bucket di directory Amazon S3

Il seguente esempio di codice mostra come iniziare a utilizzare i bucket di directory 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.

```
package com.example.s3.directorybucket;


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.model.Bucket;
import software.amazon.awssdk.services.s3.model.BucketInfo;
import software.amazon.awssdk.services.s3.model.BucketType;
import software.amazon.awssdk.services.s3.model.CreateBucketConfiguration;
import software.amazon.awssdk.services.s3.model.CreateBucketRequest;
import software.amazon.awssdk.services.s3.model.CreateBucketResponse;
import software.amazon.awssdk.services.s3.model.DataRedundancy;
import software.amazon.awssdk.services.s3.model.DeleteBucketRequest;
import software.amazon.awssdk.services.s3.model.ListDirectoryBucketsRequest;
import software.amazon.awssdk.services.s3.model.ListDirectoryBucketsResponse;
import software.amazon.awssdk.services.s3.model.LocationInfo;
import software.amazon.awssdk.services.s3.model.LocationType;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.util.List;
import java.util.stream.Collectors;

import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;

/**
 * Before running this example:
 * <p>
 * The SDK must be able to authenticate AWS requests on your behalf. If you have
```

```
 * not configured
 * authentication for SDKs and tools, see
 * https://docs.aws.amazon.com/sdkref/latest/guide/access.html in the AWS SDKs
 * and Tools Reference Guide.
 * <p>
 * You must have a runtime environment configured with the Java SDK.
 * See
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/setup.html in
 * the Developer Guide if this is not set up.
 * <p>
 * To use S3 directory buckets, configure a gateway VPC endpoint. This is the
 * recommended method to enable directory bucket traffic without
 * requiring an internet gateway or NAT device. For more information on
 * configuring VPC gateway endpoints, visit
 * https://docs.aws.amazon.com/AmazonS3/latest/userguide/s3-express-
networking.html#s3-express-networking-vpc-gateway.
 * <p>
 * Directory buckets are available in specific AWS Regions and Zones. For
 * details on Regions and Zones supporting directory buckets, see
 * https://docs.aws.amazon.com/AmazonS3/latest/userguide/s3-express-
networking.html#s3-express-endpoints.
 */

public class HelloS3DirectoryBuckets {
    private static final Logger logger =
 LoggerFactory.getLogger(HelloS3DirectoryBuckets.class);

    public static void main(String[] args) {
        String bucketName = "test-bucket-" + System.currentTimeMillis() + "--usw2-
az1--x-s3";
        Region region = Region.US_WEST_2;
        String zone = "usw2-az1";
        S3Client s3Client = createS3Client(region);

        try {
            // Create the directory bucket
            createDirectoryBucket(s3Client, bucketName, zone);
            logger.info("Created bucket: {}", bucketName);

            // List all directory buckets
            List<String> bucketNames = listDirectoryBuckets(s3Client);
            bucketNames.forEach(name -> logger.info("Bucket Name: {}", name));
        } catch (S3Exception e) {
```

```
            logger.error("An error occurred during S3 operations: {} - Error code:
{}",
                    e.awsErrorDetails().errorMessage(),
e.awsErrorDetails().errorCode(), e);
        } finally {
            try {
                // Delete the created bucket
                deleteDirectoryBucket(s3Client, bucketName);
                logger.info("Deleted bucket: {}", bucketName);
            } catch (S3Exception e) {
                logger.error("Failed to delete the bucket due to S3 error: {} -
Error code: {}",
                        e.awsErrorDetails().errorMessage(),
e.awsErrorDetails().errorCode(), e);
            } catch (RuntimeException e) {
                logger.error("Failed to delete the bucket due to unexpected error:
{}", e.getMessage(), e);
            } finally {
                s3Client.close();
            }
        }
    }

    /**
     * Creates a new S3 directory bucket in a specified Zone (For example, a
     * specified Availability Zone in this code example).
     *
     * @param s3Client   The S3 client used to create the bucket
     * @param bucketName The name of the bucket to be created
     * @param zone       The region where the bucket will be created
     * @throws S3Exception if there's an error creating the bucket
     */
    public static void createDirectoryBucket(S3Client s3Client, String bucketName,
String zone) throws S3Exception {
        logger.info("Creating bucket: {}", bucketName);

        CreateBucketConfiguration bucketConfiguration =
CreateBucketConfiguration.builder()
                .location(LocationInfo.builder()
                        .type(LocationType.AVAILABILITY_ZONE)
                        .name(zone).build())
                .bucket(BucketInfo.builder()
                        .type(BucketType.DIRECTORY)
                        .dataRedundancy(DataRedundancy.SINGLE_AVAILABILITY_ZONE)
```

```
                        .build())
                .build();
        try {
            CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
                    .bucket(bucketName)
                    .createBucketConfiguration(bucketConfiguration).build();
            CreateBucketResponse response = s3Client.createBucket(bucketRequest);
            logger.info("Bucket created successfully with location: {}",
response.location());
        } catch (S3Exception e) {
            logger.error("Error creating bucket: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode(), e);
            throw e;
        }
    }

    /**
     * Lists all S3 directory buckets.
     *
     * @param s3Client The S3 client used to interact with S3
     * @return A list of bucket names
     */
    public static List<String> listDirectoryBuckets(S3Client s3Client) {
        logger.info("Listing all directory buckets");

        try {
            // Create a ListBucketsRequest
            ListDirectoryBucketsRequest listBucketsRequest =
ListDirectoryBucketsRequest.builder().build();

            // Retrieve the list of buckets
            ListDirectoryBucketsResponse response =
s3Client.listDirectoryBuckets(listBucketsRequest);

            // Extract bucket names
            List<String> bucketNames = response.buckets().stream()
                    .map(Bucket::name)
                    .collect(Collectors.toList());

            return bucketNames;
        } catch (S3Exception e) {
            logger.error("Failed to list buckets: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
```

```
                    e.awsErrorDetails().errorCode(), e);
            throw e;
        }
    }

    /**
     * Deletes the specified S3 directory bucket.
     *
     * @param s3Client   The S3 client used to interact with S3
     * @param bucketName The name of the bucket to delete
     */
    public static void deleteDirectoryBucket(S3Client s3Client, String bucketName) {
        try {
            DeleteBucketRequest deleteBucketRequest = DeleteBucketRequest.builder()
                    .bucket(bucketName)
                    .build();
            s3Client.deleteBucket(deleteBucketRequest);
        } catch (S3Exception e) {
            logger.error("Failed to delete bucket: " + bucketName + " - Error code:
 " + e.awsErrorDetails().errorCode(),
                    e);
            throw e;
        }
    }

}
```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella Documentazione di riferimento delle API AWS SDK for Java 2.x .
  - CreateBucket
  - ListDirectoryBuckets

Argomenti

- Nozioni di base
- Azioni
- Scenari

Nozioni di base

Informazioni di base

L'esempio di codice seguente mostra come:

- Configura un VPC e un endpoint VPC.
- Configura le politiche, i ruoli e l'utente per utilizzare i bucket di directory S3 e la classe di storage S3 Express One Zone.
- Crea due client S3.
- Creare due bucket.
- Crea un oggetto e copialo.
- Dimostra la differenza di prestazioni.
- Compila i bucket per mostrare la differenza lessicografica.
- Chiedi all'utente di vedere se desidera ripulire 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.

Esegui uno scenario interattivo che dimostri le caratteristiche di Amazon S3.

```
public class S3DirectoriesScenario {

    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    private static final Logger logger =
 LoggerFactory.getLogger(S3DirectoriesScenario.class);
    static Scanner scanner = new Scanner(System.in);

    private static S3AsyncClient mS3RegularClient;
    private static S3AsyncClient mS3ExpressClient;

    private static String mdirectoryBucketName;
    private static String mregularBucketName;
```

```
   private static String stackName = "cfn-stack-s3-express-basics--" +
UUID.randomUUID();

   private static String regularUser = "";
   private static String vpcId = "";
   private static String expressUser = "";

   private static String vpcEndpointId = "";

   private static final S3DirectoriesActions s3DirectoriesActions = new
S3DirectoriesActions();

   public static void main(String[] args) {
       try {
           s3ExpressScenario();
       } catch (RuntimeException e) {
           logger.info(e.getMessage());
       }
   }

   // Runs the scenario.
   private static void s3ExpressScenario() {
       logger.info(DASHES);
       logger.info("Welcome to the Amazon S3 Express Basics demo using AWS SDK for
Java V2.");
       logger.info("""
           Let's get started! First, please note that S3 Express One Zone works
best when working within the AWS infrastructure,
           specifically when working in the same Availability Zone (AZ). To see the
best results in this example and when you implement
           directory buckets into your infrastructure, it is best to put your
compute resources in the same AZ as your directory
           bucket.
           """);
       waitForInputToContinue(scanner);
       logger.info(DASHES);

       // Create an optional VPC and create 2 IAM users.
       UserNames userNames = createVpcUsers();
       String expressUserName = userNames.getExpressUserName();
       String regularUserName = userNames.getRegularUserName();

       //  Set up two S3 clients, one regular and one express,
```

```
        //  and two buckets, one regular and one directory.
        setupClientsAndBuckets(expressUserName, regularUserName);

        // Create an S3 session for the express S3 client and add objects to the
buckets.
        logger.info("Now let's add some objects to our buckets and demonstrate how
to work with S3 Sessions.");
        waitForInputToContinue(scanner);
        String bucketObject = createSessionAddObjects();

        // Demonstrate performance differences between regular and directory
buckets.
        demonstratePerformance(bucketObject);

        // Populate the buckets to show the lexicographical difference between
        // regular and express buckets.
        showLexicographicalDifferences(bucketObject);

        logger.info(DASHES);
        logger.info("That's it for our tour of the basic operations for S3 Express
One Zone.");
        logger.info("Would you like to cleanUp the AWS resources? (y/n): ");
        String response = scanner.next().trim().toLowerCase();
        if (response.equals("y")) {
            cleanUp(stackName);
        }
    }

    /*
      Delete resources created by this scenario.
    */
    public static void cleanUp(String stackName) {
        try {
            if (mdirectoryBucketName != null) {
                s3DirectoriesActions.deleteBucketAndObjectsAsync(mS3ExpressClient,
mdirectoryBucketName).join();
            }
            logger.info("Deleted directory bucket " + mdirectoryBucketName);
            mdirectoryBucketName = null;
            if (mregularBucketName != null) {
                s3DirectoriesActions.deleteBucketAndObjectsAsync(mS3RegularClient,
mregularBucketName).join();
            }
        } catch (CompletionException ce) {
```

```
            Throwable cause = ce.getCause();
            if (cause instanceof S3Exception) {
                logger.error("S3Exception occurred: {}", cause.getMessage(), ce);
            } else {
                logger.error("An unexpected error occurred: {}", cause.getMessage(),
ce);
            }
        }

        logger.info("Deleted regular bucket " + mregularBucketName);
        mregularBucketName = null;
        CloudFormationHelper.destroyCloudFormationStack(stackName);
    }

    private static void showLexicographicalDifferences(String bucketObject) {
        logger.info(DASHES);
        logger.info("""
            7. Populate the buckets to show the lexicographical (alphabetical)
difference
            when object names are listed. Now let's explore how directory buckets
store
            objects in a different manner to regular buckets. The key is in the
name
            "Directory". Where regular buckets store their key/value pairs in a
            flat manner, directory buckets use actual directories/folders.
            This allows for more rapid indexing, traversing, and therefore
            retrieval times!

            The more segmented your bucket is, with lots of
            directories, sub-directories, and objects, the more efficient it
becomes.
            This structural difference also causes `ListObject` operations to
behave
            differently, which can cause unexpected results. Let's add a few more
            objects in sub-directories to see how the output of
            ListObjects changes.
            """);

        waitForInputToContinue(scanner);

        //  Populate a few more files in each bucket so that we can use
        //  ListObjects and show the difference.
        String otherObject = "other/" + bucketObject;
        String altObject = "alt/" + bucketObject;
```

```
        String otherAltObject = "other/alt/" + bucketObject;

        try {
            s3DirectoriesActions.putObjectAsync(mS3RegularClient,
mregularBucketName, otherObject, "").join();
            s3DirectoriesActions.putObjectAsync(mS3ExpressClient,
mdirectoryBucketName, otherObject, "").join();
            s3DirectoriesActions.putObjectAsync(mS3RegularClient,
mregularBucketName, altObject, "").join();
            s3DirectoriesActions.putObjectAsync(mS3ExpressClient,
mdirectoryBucketName, altObject, "").join();
            s3DirectoriesActions.putObjectAsync(mS3RegularClient,
mregularBucketName, otherAltObject, "").join();
            s3DirectoriesActions.putObjectAsync(mS3ExpressClient,
mdirectoryBucketName, otherAltObject, "").join();

        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof NoSuchBucketException) {
                logger.error("S3Exception occurred: {}", cause.getMessage(), ce);
            } else {
                logger.error("An unexpected error occurred: {}", cause.getMessage(),
ce);
            }
            return;
        }

        try {
            // List objects in both S3 buckets.
            List<String> dirBucketObjects =
s3DirectoriesActions.listObjectsAsync(mS3ExpressClient,
mdirectoryBucketName).join();
            List<String> regBucketObjects =
s3DirectoriesActions.listObjectsAsync(mS3RegularClient, mregularBucketName).join();

            logger.info("Directory bucket content");
            for (String obj : dirBucketObjects) {
                logger.info(obj);
            }

            logger.info("Regular bucket content");
            for (String obj : regBucketObjects) {
                logger.info(obj);
            }
```

```
        } catch (CompletionException e) {
            logger.error("Async operation failed: {} ", e.getCause().getMessage());
            return;
        }

        logger.info("""
            Notice how the regular bucket lists objects in lexicographical order,
while the directory bucket does not. This is
            because the regular bucket considers the whole "key" to be the object
identifier, while the directory bucket actually
            creates directories and uses the object "key" as a path to the object.
            """);
        waitForInputToContinue(scanner);
    }

    /**
     * Demonstrates the performance difference between downloading an object from a
directory bucket and a regular bucket.
     *
     * <p>This method:
     * <ul>
     *     <li>Prompts the user to choose the number of downloads (default is
1,000).</li>
     *     <li>Downloads the specified object from the directory bucket and measures
the total time.</li>
     *     <li>Downloads the same object from the regular bucket and measures the
total time.</li>
     *     <li>Compares the time differences and prints the results.</li>
     * </ul>
     *
     * <p>Note: The performance difference will be more pronounced if this example
is run on an EC2 instance
     * in the same Availability Zone as the buckets.
     *
     * @param bucketObject the name of the object to download
     */
    private static void demonstratePerformance(String bucketObject) {
        logger.info(DASHES);
        logger.info("6. Demonstrate the performance difference.");
        logger.info("""
            Now, let's do a performance test. We'll download the same object from
each
            bucket repeatedly and compare the total time needed.
```

```
            Note: the performance difference will be much more pronounced if this
            example is run in an EC2 instance in the same Availability Zone as
            the bucket.
            """);
        waitForInputToContinue(scanner);

        int downloads = 1000; // Default value.
        logger.info("The default number of downloads of the same object for this
example is set at " + downloads + ".");

        // Ask if the user wants to download a different number.
        logger.info("Would you like to download the file a different number of
times? (y/n): ");
        String response = scanner.next().trim().toLowerCase();
        if (response.equals("y")) {
            int maxDownloads = 1_000_000;

            // Ask for a valid number of downloads.
            while (true) {
                logger.info("Enter a number between 1 and " + maxDownloads + " for
the number of downloads: ");
                if (scanner.hasNextInt()) {
                    downloads = scanner.nextInt();
                    if (downloads >= 1 && downloads <= maxDownloads) {
                        break;
                    } else {
                        logger.info("Please enter a number between 1 and " +
maxDownloads + ".");
                    }
                } else {
                    logger.info("Invalid input. Please enter a valid integer.");
                    scanner.next();
                }
            }

            logger.info("You have chosen to download {}  items.", downloads);
        } else {
            logger.info("No changes made. Using default downloads: {}", downloads);
        }
        // Simulating the download process for the directory bucket.
        logger.info("Downloading from the directory bucket.");
        long directoryTimeStart = System.nanoTime();
        for (int index = 0; index < downloads; index++) {
            if (index % 50 == 0) {
```

```
                logger.info("Download " + index + " of " + downloads);
            }

            try {
                // Get the object from the directory bucket.
                s3DirectoriesActions.getObjectAsync(mS3ExpressClient,
mdirectoryBucketName, bucketObject).join();
            } catch (CompletionException ce) {
                Throwable cause = ce.getCause();
                if (cause instanceof NoSuchKeyException) {
                    logger.error("S3Exception occurred: {}", cause.getMessage(),
ce);
                } else {
                    logger.error("An unexpected error occurred: {}",
cause.getMessage(), ce);
                }
                return;
            }
        }
        long directoryTimeDifference = System.nanoTime() - directoryTimeStart;

        // Download from the regular bucket.
        logger.info("Downloading from the regular bucket.");
        long normalTimeStart = System.nanoTime();
        for (int index = 0; index < downloads; index++) {
            if (index % 50 == 0) {
                logger.info("Download " + index + " of " + downloads);
            }

            try {
                s3DirectoriesActions.getObjectAsync(mS3RegularClient,
mregularBucketName, bucketObject).join();
            } catch (CompletionException ce) {
                Throwable cause = ce.getCause();
                if (cause instanceof NoSuchKeyException) {
                    logger.error("S3Exception occurred: {}", cause.getMessage(),
ce);
                } else {
                    logger.error("An unexpected error occurred: {}",
cause.getMessage(), ce);
                }
                return;
            }
        }
```

```
        long normalTimeDifference = System.nanoTime() - normalTimeStart;
        logger.info("The directory bucket took " + directoryTimeDifference
+ " nanoseconds, while the regular bucket took " + normalTimeDifference + "
nanoseconds.");
        long difference = normalTimeDifference - directoryTimeDifference;
        logger.info("That's a difference of " + difference + " nanoseconds, or");
        logger.info(difference / 1_000_000_000.0 + " seconds.");

        if (difference < 0) {
            logger.info("The directory buckets were slower. This can happen if you
are not running on the cloud within a VPC.");
        }
        waitForInputToContinue(scanner);
    }

    private static String createSessionAddObjects() {
        logger.info(DASHES);
        logger.info("""
            5. Create an object and copy it.
            We'll create an object consisting of some text and upload it to the
            regular bucket.
            """);
        waitForInputToContinue(scanner);

        String bucketObject = "basic-text-object.txt";
        try {
            s3DirectoriesActions.putObjectAsync(mS3RegularClient,
mregularBucketName, bucketObject, "Look Ma, I'm a bucket!").join();
            s3DirectoriesActions.createSessionAsync(mS3ExpressClient,
mdirectoryBucketName).join();

            // Copy the object to the destination S3 bucket.
            s3DirectoriesActions.copyObjectAsync(mS3ExpressClient,
mregularBucketName, bucketObject, mdirectoryBucketName, bucketObject).join();
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof S3Exception) {
                logger.error("S3Exception occurred: {}", cause.getMessage(), ce);
            } else {
                logger.error("An unexpected error occurred: {}", cause.getMessage(),
ce);
            }
        }
```

```
       logger.info("""
           It worked! This is because the S3Client that performed the copy
operation
           is the expressClient using the credentials for the user with permission
to
           work with directory buckets.

           It's important to remember the user permissions when interacting with
           directory buckets. Instead of validating permissions on every call as
           regular buckets do, directory buckets utilize the user credentials and
session
           token to validate. This allows for much faster connection speeds on
every call.
           For single calls, this is low, but for many concurrent calls
           this adds up to a lot of time saved.
           """);
       waitForInputToContinue(scanner);
       return bucketObject;
   }

   /**
    * Creates VPC users for the S3 Express One Zone scenario.
    * <p>
    * This method performs the following steps:
    * <ol>
    *     <li>Optionally creates a new VPC and VPC Endpoint if the application
is running in an EC2 instance in the same Availability Zone as the directory
buckets.</li>
    *     <li>Creates two IAM users: one with S3 Express One Zone permissions and
one without.</li>
    * </ol>
    *
    * @return a {@link UserNames} object containing the names of the created IAM
users
    */
   public static UserNames createVpcUsers() {
       /*
       Optionally create a VPC.
       Create two IAM users, one with S3 Express One Zone permissions and one
without.
       */
       logger.info(DASHES);
       logger.info("""
```

```
            1. First, we'll set up a new VPC and VPC Endpoint if this program is
running in an EC2 instance in the same AZ as your\s
            directory buckets will be. Are you running this in an EC2 instance
located in the same AZ as your intended directory buckets?
            """);

        logger.info("Do you want to setup a VPC Endpoint? (y/n)");
        String endpointAns = scanner.nextLine().trim();
        if (endpointAns.equalsIgnoreCase("y")) {
            logger.info("""
                Great! Let's set up a VPC, retrieve the Route Table from it, and
create a VPC Endpoint to connect the S3 Client to.
                """);
            try {
                s3DirectoriesActions.setupVPCAsync().join();
            } catch (CompletionException ce) {
                Throwable cause = ce.getCause();
                if (cause instanceof Ec2Exception) {
                    logger.error("IamException occurred: {}", cause.getMessage(),
ce);
                } else {
                    logger.error("An unexpected error occurred: {}",
cause.getMessage(), ce);
                }
            }
            waitForInputToContinue(scanner);
        } else {
            logger.info("Skipping the VPC setup. Don't forget to use this in
production!");
        }
        logger.info(DASHES);
        logger.info("""
            2. Create a RegularUser and ExpressUser by using the AWS CDK.
            One IAM User, named RegularUser, will have permissions to work only
            with regular buckets and one IAM user, named ExpressUser, will have
            permissions to work only with directory buckets.
            """);
        waitForInputToContinue(scanner);

        // Create two users required for this scenario.
        Map<String, String> stackOutputs = createUsersUsingCDK(stackName);
        regularUser = stackOutputs.get("RegularUser");
        expressUser = stackOutputs.get("ExpressUser");
```

```
        UserNames names = new UserNames();
        names.setRegularUserName(regularUser);
        names.setExpressUserName(expressUser);
        return names;
    }

    /**
     * Creates users using AWS CloudFormation.
     *
     * @return a {@link Map} of String keys and String values representing the stack
outputs,
     * which may include user-related information such as user names and IDs.
     */
    public static Map<String, String> createUsersUsingCDK(String stackName) {
        logger.info("We'll use an AWS CloudFormation template to create the IAM
users and policies.");
        CloudFormationHelper.deployCloudFormationStack(stackName);
        return CloudFormationHelper.getStackOutputsAsync(stackName).join();
    }

    /**
     * Sets up the necessary clients and buckets for the S3 Express service.
     *
     * @param expressUserName the username for the user with S3 Express permissions
     * @param regularUserName the username for the user with regular S3 permissions
     */
    public static void setupClientsAndBuckets(String expressUserName, String
regularUserName) {
        Scanner locscanner = new Scanner(System.in);
        String accessKeyIdforRegUser;
        String secretAccessforRegUser;
        try {
            CreateAccessKeyResponse keyResponse =
s3DirectoriesActions.createAccessKeyAsync(regularUserName).join();
            accessKeyIdforRegUser = keyResponse.accessKey().accessKeyId();
            secretAccessforRegUser = keyResponse.accessKey().secretAccessKey();
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof IamException) {
                logger.error("IamException occurred: {}", cause.getMessage(), ce);
            } else {
                logger.error("An unexpected error occurred: {}", cause.getMessage(),
ce);
            }
```

```
            return;
        }

        String accessKeyIdforExpressUser;
        String secretAccessforExpressUser;
        try {
            CreateAccessKeyResponse keyResponseExpress =
s3DirectoriesActions.createAccessKeyAsync(expressUserName).join();
            accessKeyIdforExpressUser =
keyResponseExpress.accessKey().accessKeyId();
            secretAccessforExpressUser =
keyResponseExpress.accessKey().secretAccessKey();
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof IamException) {
                logger.error("IamException occurred: {}", cause.getMessage(), ce);
            } else {
                logger.error("An unexpected error occurred: {}", cause.getMessage(),
ce);
            }
            return;
        }

        logger.info(DASHES);
        logger.info("""
            3. Create two S3Clients; one uses the ExpressUser's credentials and one
uses the RegularUser's credentials.
            The 2 S3Clients will use different credentials.
            """);
        waitForInputToContinue(locscanner);
        try {
            mS3RegularClient =
createS3ClientWithAccessKeyAsync(accessKeyIdforRegUser,
secretAccessforRegUser).join();
            mS3ExpressClient =
createS3ClientWithAccessKeyAsync(accessKeyIdforExpressUser,
secretAccessforExpressUser).join();
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof IllegalArgumentException) {
                logger.error("An invalid argument exception occurred: {}",
cause.getMessage(), ce);
            } else {
```

```
                logger.error("An unexpected error occurred: {}", cause.getMessage(),
ce);
            }
            return;
        }

        logger.info("""
            We can now use the ExpressUser client to make calls to S3 Express
operations.
            """);
        waitForInputToContinue(locscanner);
        logger.info(DASHES);
        logger.info("""
            4. Create two buckets.
            Now we will create a directory bucket which is the linchpin of the S3
Express One Zone service. Directory buckets
            behave differently from regular S3 buckets which we will explore here.
We'll also create a regular bucket, put
            an object into the regular bucket, and copy it to the directory bucket.
            """);

        logger.info("""
            Now, let's choose an availability zone (AZ) for the directory bucket.
            We'll choose one that is supported.
            """);
        String zoneId;
        String regularBucketName;
        try {
            zoneId = s3DirectoriesActions.selectAvailabilityZoneIdAsync().join();
            regularBucketName = "reg-bucket-" + System.currentTimeMillis();
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof Ec2Exception) {
                logger.error("EC2Exception occurred: {}", cause.getMessage(), ce);
            } else {
                logger.error("An unexpected error occurred: {}", cause.getMessage(),
ce);
            }
            return;
        }
        logger.info("""
            Now, let's create the actual directory bucket, as well as a regular
bucket."
             """);
```

```
        String directoryBucketName = "test-bucket-" + System.currentTimeMillis() +
"--" + zoneId + "--x-s3";
        try {
            s3DirectoriesActions.createDirectoryBucketAsync(mS3ExpressClient,
directoryBucketName, zoneId).join();
            logger.info("Created directory bucket {}", directoryBucketName);
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof BucketAlreadyExistsException) {
                logger.error("The bucket already exists. Moving on: {}",
cause.getMessage(), ce);
            } else {
                logger.error("An unexpected error occurred: {}", cause.getMessage(),
ce);
                return;
            }
        }

        // Assign to the data member.
        mdirectoryBucketName = directoryBucketName;
        try {
            s3DirectoriesActions.createBucketAsync(mS3RegularClient,
regularBucketName).join();
            logger.info("Created regular bucket {} ", regularBucketName);
            mregularBucketName = regularBucketName;
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof BucketAlreadyExistsException) {
                logger.error("The bucket already exists. Moving on: {}",
cause.getMessage(), ce);
            } else {
                logger.error("An unexpected error occurred: {}", cause.getMessage(),
ce);
                return;
            }
        }
        logger.info("Great! Both buckets were created.");
        waitForInputToContinue(locscanner);
    }

    /**
     * Creates an asynchronous S3 client with the specified access key and secret
access key.
```

```
     *
     * @param accessKeyId     the AWS access key ID
     * @param secretAccessKey the AWS secret access key
     * @return a {@link CompletableFuture} that asynchronously creates the S3 client
     * @throws IllegalArgumentException if the access key ID or secret access key is
 null
     */
    public static CompletableFuture<S3AsyncClient>
 createS3ClientWithAccessKeyAsync(String accessKeyId, String secretAccessKey) {
        return CompletableFuture.supplyAsync(() -> {
            // Validate input parameters
            if (accessKeyId == null || accessKeyId.isBlank() || secretAccessKey ==
 null || secretAccessKey.isBlank()) {
                throw new IllegalArgumentException("Access Key ID and Secret Access
 Key must not be null or empty");
            }

            AwsBasicCredentials awsCredentials =
 AwsBasicCredentials.create(accessKeyId, secretAccessKey);
            return S3AsyncClient.builder()

 .credentialsProvider(StaticCredentialsProvider.create(awsCredentials))
                .region(Region.US_WEST_2)
                .build();
        });
    }

    private static void waitForInputToContinue(Scanner scanner) {
        while (true) {
            logger.info("");
            logger.info("Enter 'c' followed by <ENTER> to continue:");
            String input = scanner.nextLine();

            if (input.trim().equalsIgnoreCase("c")) {
                logger.info("Continuing with the program...");
                logger.info("");
                break;
            } else {
                logger.info("Invalid input. Please try again.");
            }
        }
    }
}
```

Una classe wrapper per i metodi SDK di Amazon S3.

```
public class S3DirectoriesActions {

    private static IamAsyncClient iamAsyncClient;

    private static Ec2AsyncClient ec2AsyncClient;
    private static final Logger logger =
 LoggerFactory.getLogger(S3DirectoriesActions.class);

    private static IamAsyncClient getIAMAsyncClient() {
        if (iamAsyncClient == null) {
            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                .maxConcurrency(100)
                .connectionTimeout(Duration.ofSeconds(60))
                .readTimeout(Duration.ofSeconds(60))
                .writeTimeout(Duration.ofSeconds(60))
                .build();

            ClientOverrideConfiguration overrideConfig =
 ClientOverrideConfiguration.builder()
                .apiCallTimeout(Duration.ofMinutes(2))
                .apiCallAttemptTimeout(Duration.ofSeconds(90))
                .retryStrategy(RetryMode.STANDARD)
                .build();

            iamAsyncClient = IamAsyncClient.builder()
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return iamAsyncClient;
    }

    private static Ec2AsyncClient getEc2AsyncClient() {
        if (ec2AsyncClient == null) {
            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                .maxConcurrency(100)
                .connectionTimeout(Duration.ofSeconds(60))
                .readTimeout(Duration.ofSeconds(60))
                .writeTimeout(Duration.ofSeconds(60))
```

```
                .build();

            ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
                .apiCallTimeout(Duration.ofMinutes(2))
                .apiCallAttemptTimeout(Duration.ofSeconds(90))
                .retryStrategy(RetryMode.STANDARD)
                .build();

            ec2AsyncClient = Ec2AsyncClient.builder()
                .httpClient(httpClient)
                .region(Region.US_WEST_2)
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return ec2AsyncClient;
    }

    /**
     * Deletes the specified S3 bucket and all the objects within it asynchronously.
     *
     * @param s3AsyncClient the S3 asynchronous client to use for the operations
     * @param bucketName the name of the S3 bucket to be deleted
     * @return a {@link CompletableFuture} that completes with a {@link
WaiterResponse} containing the
     *         {@link HeadBucketResponse} when the bucket has been successfully
deleted
     * @throws CompletionException if there was an error deleting the bucket or its
objects
     */
    public CompletableFuture<WaiterResponse<HeadBucketResponse>>
deleteBucketAndObjectsAsync(S3AsyncClient s3AsyncClient, String bucketName) {
        ListObjectsV2Request listRequest = ListObjectsV2Request.builder()
            .bucket(bucketName)
            .build();

        return s3AsyncClient.listObjectsV2(listRequest)
            .thenCompose(listResponse -> {
                if (!listResponse.contents().isEmpty()) {
                    List<ObjectIdentifier> objectIdentifiers =
listResponse.contents().stream()
                        .map(s3Object ->
ObjectIdentifier.builder().key(s3Object.key()).build())
                        .collect(Collectors.toList());
```

```
                    DeleteObjectsRequest deleteRequest =
DeleteObjectsRequest.builder()
                        .bucket(bucketName)
                        .delete(Delete.builder().objects(objectIdentifiers).build())
                        .build();

                    return s3AsyncClient.deleteObjects(deleteRequest)
                        .thenAccept(deleteResponse -> {
                            if (!deleteResponse.errors().isEmpty()) {
                                deleteResponse.errors().forEach(error ->
                                    logger.error("Couldn't delete object " +
error.key() + ". Reason: " + error.message()));
                            }
                        });
                }
                return CompletableFuture.completedFuture(null);
            })
            .thenCompose(ignored -> {
                DeleteBucketRequest deleteBucketRequest =
DeleteBucketRequest.builder()
                    .bucket(bucketName)
                    .build();
                return s3AsyncClient.deleteBucket(deleteBucketRequest);
            })
            .thenCompose(ignored -> {
                S3AsyncWaiter waiter = s3AsyncClient.waiter();
                HeadBucketRequest headBucketRequest =
HeadBucketRequest.builder().bucket(bucketName).build();
                return waiter.waitUntilBucketNotExists(headBucketRequest);
            })
            .whenComplete((ignored, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof S3Exception) {
                        throw new CompletionException("Error deleting bucket: " +
bucketName, cause);
                    }
                    throw new CompletionException("Failed to delete bucket and
objects: " + bucketName, exception);
                }
                logger.info("Bucket deleted successfully: " + bucketName);
            });
    }
```

```
   /**
    *  Lists the objects in an S3 bucket asynchronously.
    *
    * @param s3Client the S3 async client to use for the operation
    * @param bucketName the name of the S3 bucket containing the objects to list
    * @return a {@link CompletableFuture} that contains the list of object keys in
the specified bucket
    */
   public CompletableFuture<List<String>> listObjectsAsync(S3AsyncClient s3Client,
String bucketName) {
       ListObjectsV2Request request = ListObjectsV2Request.builder()
           .bucket(bucketName)
           .build();

       return s3Client.listObjectsV2(request)
           .thenApply(response -> response.contents().stream()
               .map(S3Object::key)
               .toList())
           .whenComplete((result, exception) -> {
               if (exception != null) {
                   throw new CompletionException("Couldn't list objects in bucket:
" + bucketName, exception);
               }
           });
   }

   /**
    * Retrieves an object from an Amazon S3 bucket asynchronously.
    *
    * @param s3Client   the S3 async client to use for the operation
    * @param bucketName the name of the S3 bucket containing the object
    * @param keyName    the unique identifier (key) of the object to retrieve
    * @return a {@link CompletableFuture} that, when completed, contains the
object's content as a {@link ResponseBytes} of {@link GetObjectResponse}
    */
   public CompletableFuture<ResponseBytes<GetObjectResponse>>
getObjectAsync(S3AsyncClient s3Client, String bucketName, String keyName) {
       GetObjectRequest objectRequest = GetObjectRequest.builder()
           .key(keyName)
           .bucket(bucketName)
           .build();

       // Get the object asynchronously and transform it into a byte array
```

```
        return s3Client.getObject(objectRequest, AsyncResponseTransformer.toBytes())
            .exceptionally(exception -> {
                Throwable cause = exception.getCause();
                if (cause instanceof NoSuchKeyException) {
                    throw new CompletionException("Failed to get the object. Reason:
" + ((S3Exception) cause).awsErrorDetails().errorMessage(), cause);
                }
                throw new CompletionException("Failed to get the object",
exception);
            });
    }

    /**
     * Asynchronously copies an object from one S3 bucket to another.
     *
     * @param s3Client           the S3 async client to use for the copy operation
     * @param sourceBucket       the name of the source bucket
     * @param sourceKey          the key of the object to be copied in the source
bucket
     * @param destinationBucket  the name of the destination bucket
     * @param destinationKey     the key of the copied object in the destination
bucket
     * @return a {@link CompletableFuture} that completes when the copy operation is
finished
     */
    public CompletableFuture<Void> copyObjectAsync(S3AsyncClient s3Client, String
sourceBucket, String sourceKey, String destinationBucket, String destinationKey) {
        CopyObjectRequest copyRequest = CopyObjectRequest.builder()
            .sourceBucket(sourceBucket)
            .sourceKey(sourceKey)
            .destinationBucket(destinationBucket)
            .destinationKey(destinationKey)
            .build();

        return s3Client.copyObject(copyRequest)
            .thenRun(() -> logger.info("Copied object '" + sourceKey + "' from
bucket '" + sourceBucket + "' to bucket '" + destinationBucket + "'"))
            .whenComplete((ignored, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof S3Exception) {
                        throw new CompletionException("Couldn't copy object '" +
sourceKey + "' from bucket '" + sourceBucket + "' to bucket '" + destinationBucket
+ "'. Reason: " + ((S3Exception) cause).awsErrorDetails().errorMessage(), cause);
```

```
                    }
                    throw new CompletionException("Failed to copy object",
exception);
                }
            });
    }

    /**
     * Asynchronously creates a session for the specified S3 bucket.
     *
     * @param s3Client   the S3 asynchronous client to use for creating the session
     * @param bucketName the name of the S3 bucket for which to create the session
     * @return a {@link CompletableFuture} that completes when the session is
created, or throws a {@link CompletionException} if an error occurs
     */
    public CompletableFuture<CreateSessionResponse> createSessionAsync(S3AsyncClient
s3Client, String bucketName) {
        CreateSessionRequest request = CreateSessionRequest.builder()
            .bucket(bucketName)
            .build();

        return s3Client.createSession(request)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof S3Exception) {
                        throw new CompletionException("Couldn't create the session.
Reason: " + ((S3Exception) cause).awsErrorDetails().errorMessage(), cause);
                    }
                    throw new CompletionException("Unexpected error occurred while
creating session", exception);
                }
                logger.info("Created session for bucket: " + bucketName);
            });

    }

    /**
     * Creates a new S3 directory bucket in a specified Zone (For example, a
     * specified Availability Zone in this code example).
     *
     * @param s3Client   The asynchronous S3 client used to create the bucket
     * @param bucketName The name of the bucket to be created
     * @param zone       The Availability Zone where the bucket will be created
```

```
     * @throws CompletionException if there's an error creating the bucket
     */
    public CompletableFuture<CreateBucketResponse>
createDirectoryBucketAsync(S3AsyncClient s3Client, String bucketName, String zone)
{
        logger.info("Creating bucket: " + bucketName);

        CreateBucketConfiguration bucketConfiguration =
CreateBucketConfiguration.builder()
            .location(LocationInfo.builder()
                .type(LocationType.AVAILABILITY_ZONE)
                .name(zone)
                .build())
            .bucket(BucketInfo.builder()
                .type(BucketType.DIRECTORY)
                .dataRedundancy(DataRedundancy.SINGLE_AVAILABILITY_ZONE)
                .build())
            .build();

        CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
            .bucket(bucketName)
            .createBucketConfiguration(bucketConfiguration)
            .build();

        return s3Client.createBucket(bucketRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof BucketAlreadyExistsException) {
                        throw new CompletionException("The bucket already exists: "
+ ((S3Exception) cause).awsErrorDetails().errorMessage(), cause);
                    }
                    throw new CompletionException("Unexpected error occurred while
creating bucket", exception);
                }
                logger.info("Bucket created successfully with location: " +
response.location());
            });
    }

    /**
     * Creates an S3 bucket asynchronously.
     *
     * @param s3Client    the S3 async client to use for the bucket creation
```

```
    * @param bucketName  the name of the S3 bucket to create
    * @return a {@link CompletableFuture} that completes with the {@link
WaiterResponse} containing the {@link HeadBucketResponse}
    *         when the bucket is successfully created
    * @throws CompletionException if there's an error creating the bucket
    */
   public CompletableFuture<WaiterResponse<HeadBucketResponse>>
createBucketAsync(S3AsyncClient s3Client, String bucketName) {
       CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
           .bucket(bucketName)
           .build();

       return s3Client.createBucket(bucketRequest)
           .thenCompose(response -> {
               S3AsyncWaiter s3Waiter = s3Client.waiter();
               HeadBucketRequest bucketRequestWait = HeadBucketRequest.builder()
                   .bucket(bucketName)
                   .build();
               return s3Waiter.waitUntilBucketExists(bucketRequestWait);
           })
           .whenComplete((response, exception) -> {
               if (exception != null) {
                   Throwable cause = exception.getCause();
                   if (cause instanceof BucketAlreadyExistsException) {
                       throw new CompletionException("The S3 bucket exists: " +
cause.getMessage(), cause);
                   } else {
                       throw new CompletionException("Failed to create access key:
" + exception.getMessage(), exception);
                   }
               }
               logger.info(bucketName + " is ready");
           });
   }

   /**
    * Uploads an object to an Amazon S3 bucket asynchronously.
    *
    * @param s3Client     the S3 async client to use for the upload
    * @param bucketName   the destination S3 bucket name
    * @param bucketObject the name of the object to be uploaded
    * @param text         the content to be uploaded as the object
    */
```

```
   public CompletableFuture<PutObjectResponse> putObjectAsync(S3AsyncClient
s3Client, String bucketName, String bucketObject, String text) {
       PutObjectRequest objectRequest = PutObjectRequest.builder()
           .bucket(bucketName)
           .key(bucketObject)
           .build();

       return s3Client.putObject(objectRequest, AsyncRequestBody.fromString(text))
           .whenComplete((response, exception) -> {
               if (exception != null) {
                   Throwable cause = exception.getCause();
                   if (cause instanceof NoSuchBucketException) {
                       throw new CompletionException("The S3 bucket does not exist:
" + cause.getMessage(), cause);
                   } else {
                       throw new CompletionException("Failed to create access key:
" + exception.getMessage(), exception);
                   }
               }
           });
   }

   /**
    * Creates an AWS IAM access key asynchronously for the specified user name.
    *
    * @param userName the name of the IAM user for whom to create the access key
    * @return a {@link CompletableFuture} that completes with the {@link
CreateAccessKeyResponse} containing the created access key
    */
   public CompletableFuture<CreateAccessKeyResponse> createAccessKeyAsync(String
userName) {
       CreateAccessKeyRequest request = CreateAccessKeyRequest.builder()
           .userName(userName)
           .build();

       return getIAMAsyncClient().createAccessKey(request)
           .whenComplete((response, exception) -> {
               if (response != null) {
                   logger.info("Access Key Created.");
               } else {
                   if (exception == null) {
                       Throwable cause = exception.getCause();
                       if (cause instanceof IamException) {
```

```
                            throw new CompletionException("IAM error while creating
 access key: " + cause.getMessage(), cause);
                        } else {
                            throw new CompletionException("Failed to create access
 key: " + exception.getMessage(), exception);
                        }
                    }
                }
            });
    }

    /**
     * Asynchronously selects an Availability Zone ID from the available EC2 zones.
     *
     * @return A {@link CompletableFuture} that resolves to the selected
 Availability Zone ID.
     * @throws CompletionException if an error occurs during the request or
 processing.
     */
    public CompletableFuture<String> selectAvailabilityZoneIdAsync() {
        DescribeAvailabilityZonesRequest zonesRequest =
 DescribeAvailabilityZonesRequest.builder()
            .build();

        return getEc2AsyncClient().describeAvailabilityZones(zonesRequest)
            .thenCompose(response -> {
                List<AvailabilityZone> zonesList = response.availabilityZones();
                if (zonesList.isEmpty()) {
                    logger.info("No availability zones found.");
                    return CompletableFuture.completedFuture(null); // Return null
 if no zones are found
                }

                List<String> zoneIds = zonesList.stream()
                    .map(AvailabilityZone::zoneId) // Get the zoneId (e.g., "usw2-
az1")
                    .toList();

                return CompletableFuture.supplyAsync(() ->
 promptUserForZoneSelection(zonesList, zoneIds))
                    .thenApply(selectedZone -> {
                        // Return only the selected Zone ID (e.g., "usw2-az1").
                        return selectedZone.zoneId();
                    });
```

```
            })
            .whenComplete((result, exception) -> {
                if (exception == null) {
                    if (result != null) {
                        logger.info("Selected Availability Zone ID: " + result);
                    } else {
                        logger.info("No availability zone selected.");
                    }
                } else {
                    Throwable cause = exception.getCause();
                    if (cause instanceof Ec2Exception) {
                        throw new CompletionException("EC2 error while selecting
availability zone: " + cause.getMessage(), cause);
                    }
                    throw new CompletionException("Failed to select availability
zone: " + exception.getMessage(), exception);
                }
            });
    }

    /**
     * Prompts the user to select an Availability Zone from the given list.
     *
     * @param zonesList the list of Availability Zones
     * @param zoneIds the list of zone IDs
     * @return the selected Availability Zone
     */
    private static AvailabilityZone
promptUserForZoneSelection(List<AvailabilityZone> zonesList, List<String> zoneIds)
{
        Scanner scanner = new Scanner(System.in);
        int index = -1;

        while (index < 0 || index >= zoneIds.size()) {
            logger.info("Select an availability zone:");
            IntStream.range(0, zoneIds.size()).forEach(i ->
                logger.info(i + ": " + zoneIds.get(i))
            );

            logger.info("Enter the number corresponding to your choice: ");
            if (scanner.hasNextInt()) {
                index = scanner.nextInt();
            } else {
                scanner.next();
```

```
            }
        }

        AvailabilityZone selectedZone = zonesList.get(index);
        logger.info("You selected: " + selectedZone.zoneId());
        return selectedZone;
    }

    /**
     * Asynchronously sets up a new VPC, including creating the VPC, finding the
associated route table, and
     * creating a VPC endpoint for the S3 service.
     *
     * @return a {@link CompletableFuture} that, when completed, contains a
AbstractMap with the
     *         VPC ID and VPC endpoint ID.
     */
    public CompletableFuture<AbstractMap.SimpleEntry<String, String>>
setupVPCAsync() {
        String cidr = "10.0.0.0/16";
        CreateVpcRequest vpcRequest = CreateVpcRequest.builder()
            .cidrBlock(cidr)
            .build();

        return getEc2AsyncClient().createVpc(vpcRequest)
            .thenCompose(vpcResponse -> {
                String vpcId = vpcResponse.vpc().vpcId();
                logger.info("VPC Created: {}", vpcId);

                Ec2AsyncWaiter waiter = getEc2AsyncClient().waiter();
                DescribeVpcsRequest request = DescribeVpcsRequest.builder()
                    .vpcIds(vpcId)
                    .build();

                return waiter.waitUntilVpcAvailable(request)
                    .thenApply(waiterResponse -> vpcId);
            })
            .thenCompose(vpcId -> {
                Filter filter = Filter.builder()
                    .name("vpc-id")
                    .values(vpcId)
                    .build();
```

```
                DescribeRouteTablesRequest describeRouteTablesRequest =
DescribeRouteTablesRequest.builder()
                    .filters(filter)
                    .build();

                return
getEc2AsyncClient().describeRouteTables(describeRouteTablesRequest)
                    .thenApply(routeTablesResponse -> {
                        if (routeTablesResponse.routeTables().isEmpty()) {
                            throw new CompletionException("No route tables found for
VPC: " + vpcId, null);
                        }
                        String routeTableId =
routeTablesResponse.routeTables().get(0).routeTableId();
                        logger.info("Route table found: {}", routeTableId);
                        return new AbstractMap.SimpleEntry<>(vpcId, routeTableId);
                    });
            })
            .thenCompose(vpcAndRouteTable -> {
                String vpcId = vpcAndRouteTable.getKey();
                String routeTableId = vpcAndRouteTable.getValue();
                Region region =
getEc2AsyncClient().serviceClientConfiguration().region();
                String serviceName = String.format("com.amazonaws.%s.s3express",
region.id());

                CreateVpcEndpointRequest endpointRequest =
CreateVpcEndpointRequest.builder()
                    .vpcId(vpcId)
                    .routeTableIds(routeTableId)
                    .serviceName(serviceName)
                    .build();

                return getEc2AsyncClient().createVpcEndpoint(endpointRequest)
                    .thenApply(vpcEndpointResponse -> {
                        String vpcEndpointId =
vpcEndpointResponse.vpcEndpoint().vpcEndpointId();
                        logger.info("VPC Endpoint created: {}", vpcEndpointId);
                        return new AbstractMap.SimpleEntry<>(vpcId, vpcEndpointId);
                    });
            })
            .exceptionally(exception -> {
                Throwable cause = exception.getCause() != null ?
exception.getCause() : exception;
```

```
                if (cause instanceof Ec2Exception) {
                    logger.error("EC2 error during VPC setup: {}",
 cause.getMessage(), cause);
                    throw new CompletionException("EC2 error during VPC setup: " +
 cause.getMessage(), cause);
                }

                logger.error("VPC setup failed: {}", cause.getMessage(), cause);
                throw new CompletionException("VPC setup failed: " +
 cause.getMessage(), cause);
            });
    }

}
```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella Documentazione di riferimento delle API AWS SDK for Java 2.x .
  - CopyObject
  - CreateBucket
  - DeleteBucket
  - DeleteObject
  - GetObject
  - ListObjects
  - PutObject

Azioni

`AbortMultipartUpload`

Il seguente esempio di codice mostra come usare. `AbortMultipartUpload`

SDK per Java 2.x

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

Interrompi un caricamento in più parti in un bucket di directory.

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

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static
 com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucketMultipartUpload;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;



    /**
     * Aborts a specific multipart upload for the specified S3 directory bucket.
     *
     * @param s3Client   The S3 client used to interact with S3
     * @param bucketName The name of the directory bucket
     * @param objectKey  The key (name) of the object to be uploaded
     * @param uploadId   The upload ID of the multipart upload to abort
     * @return True if the multipart upload is successfully aborted, false otherwise
     */
    public static boolean abortDirectoryBucketMultipartUpload(S3Client s3Client,
 String bucketName,
            String objectKey, String uploadId) {
        logger.info("Aborting multipart upload: {} for bucket: {}", uploadId,
 bucketName);
        try {
            // Abort the multipart upload
            AbortMultipartUploadRequest abortMultipartUploadRequest =
 AbortMultipartUploadRequest.builder()
                    .bucket(bucketName)
                    .key(objectKey)
                    .uploadId(uploadId)
                    .build();

            s3Client.abortMultipartUpload(abortMultipartUploadRequest);
            logger.info("Aborted multipart upload: {} for object: {}", uploadId,
 objectKey);
            return true;
        } catch (S3Exception e) {
```

```
            logger.error("Failed to abort multipart upload: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode(), e);
            return false;
        }
    }
```

- Per i dettagli sull'API, consulta la sezione API AbortMultipartUploadReference AWS SDK for Java 2.x .

`CompleteMultipartUpload`

Il seguente esempio di codice mostra come utilizzare`CompleteMultipartUpload`.

SDK per Java 2.x

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

Completa un caricamento in più parti in un bucket di directory.

```
import com.example.s3.util.S3DirectoryBucketUtils;
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.model.CompleteMultipartUploadRequest;
import software.amazon.awssdk.services.s3.model.CompleteMultipartUploadResponse;
import software.amazon.awssdk.services.s3.model.CompletedMultipartUpload;
import software.amazon.awssdk.services.s3.model.CompletedPart;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.io.IOException;
import java.nio.file.Path;
import java.util.List;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
```

```
import static
 com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucketMultipartUpload;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static
 com.example.s3.util.S3DirectoryBucketUtils.deleteAllObjectsInDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;
import static
 com.example.s3.util.S3DirectoryBucketUtils.multipartUploadForDirectoryBucket;


   /**
    * This method completes the multipart upload request by collating all the
    * upload parts.
    *
    * @param s3Client    The S3 client used to interact with S3
    * @param bucketName  The name of the directory bucket
    * @param objectKey   The key (name) of the object to be uploaded
    * @param uploadId    The upload ID used to track the multipart upload
    * @param uploadParts The list of completed parts
    * @return True if the multipart upload is successfully completed, false
    *         otherwise
    */
   public static boolean completeDirectoryBucketMultipartUpload(S3Client s3Client,
 String bucketName, String objectKey,
           String uploadId, List<CompletedPart> uploadParts) {
       try {
           CompletedMultipartUpload completedMultipartUpload =
 CompletedMultipartUpload.builder()
                   .parts(uploadParts)
                   .build();
           CompleteMultipartUploadRequest completeMultipartUploadRequest =
 CompleteMultipartUploadRequest.builder()
                   .bucket(bucketName)
                   .key(objectKey)
                   .uploadId(uploadId)
                   .multipartUpload(completedMultipartUpload)
                   .build();

           CompleteMultipartUploadResponse response =
 s3Client.completeMultipartUpload(completeMultipartUploadRequest);
           logger.info("Multipart upload completed. ETag: {}", response.eTag());
           return true;
       } catch (S3Exception e) {
```

```
            logger.error("Failed to complete multipart upload: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode(), e);
            return false;
        }
    }
```

- Per i dettagli sull'API, consulta la sezione AWS SDK for Java 2.x API CompleteMultipartUploadReference.

`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 da un bucket di directory a un bucket di directory.

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

import java.nio.file.Path;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static
 com.example.s3.util.S3DirectoryBucketUtils.deleteAllObjectsInDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
```

```
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;
import static com.example.s3.util.S3DirectoryBucketUtils.putDirectoryBucketObject;

    /**
     * Copies an object from one S3 general purpose bucket to one S3 directory
     * bucket.
     *
     * @param s3Client     The S3 client used to interact with S3
     * @param sourceBucket The name of the source bucket
     * @param objectKey    The key (name) of the object to be copied
     * @param targetBucket The name of the target bucket
     */
    public static void copyDirectoryBucketObject(S3Client s3Client, String
 sourceBucket, String objectKey,
            String targetBucket) {
        logger.info("Copying object: {} from bucket: {} to bucket: {}", objectKey,
 sourceBucket, targetBucket);

        try {
            // Create a CopyObjectRequest
            CopyObjectRequest copyReq = CopyObjectRequest.builder()
                    .sourceBucket(sourceBucket)
                    .sourceKey(objectKey)
                    .destinationBucket(targetBucket)
                    .destinationKey(objectKey)
                    .build();

            // Copy the object
            CopyObjectResponse copyRes = s3Client.copyObject(copyReq);
            logger.info("Successfully copied {} from bucket {} into bucket {}.
 CopyObjectResponse: {}",
                    objectKey, sourceBucket, targetBucket,
 copyRes.copyObjectResult().toString());

        } catch (S3Exception e) {
            logger.error("Failed to copy object: {} - Error code: {}",
 e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode(), e);
            throw e;
        }
    }
```

- 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.

Crea un bucket di directory S3.

```
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.model.BucketInfo;
import software.amazon.awssdk.services.s3.model.BucketType;
import software.amazon.awssdk.services.s3.model.CreateBucketConfiguration;
import software.amazon.awssdk.services.s3.model.CreateBucketRequest;
import software.amazon.awssdk.services.s3.model.CreateBucketResponse;
import software.amazon.awssdk.services.s3.model.DataRedundancy;
import software.amazon.awssdk.services.s3.model.LocationInfo;
import software.amazon.awssdk.services.s3.model.LocationType;
import software.amazon.awssdk.services.s3.model.S3Exception;

import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;

    /**
     * Creates a new S3 directory bucket in a specified Zone (For example, a
     * specified Availability Zone in this code example).
     *
     * @param s3Client   The S3 client used to create the bucket
     * @param bucketName The name of the bucket to be created
     * @param zone       The region where the bucket will be created
     * @throws S3Exception if there's an error creating the bucket
     */
    public static void createDirectoryBucket(S3Client s3Client, String bucketName,
 String zone) throws S3Exception {
```

```
        logger.info("Creating bucket: {}", bucketName);

        CreateBucketConfiguration bucketConfiguration =
CreateBucketConfiguration.builder()
                .location(LocationInfo.builder()
                        .type(LocationType.AVAILABILITY_ZONE)
                        .name(zone).build())
                .bucket(BucketInfo.builder()
                        .type(BucketType.DIRECTORY)
                        .dataRedundancy(DataRedundancy.SINGLE_AVAILABILITY_ZONE)
                        .build())
                .build();
        try {
            CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
                    .bucket(bucketName)
                    .createBucketConfiguration(bucketConfiguration).build();
            CreateBucketResponse response = s3Client.createBucket(bucketRequest);
            logger.info("Bucket created successfully with location: {}",
response.location());
        } catch (S3Exception e) {
            logger.error("Error creating bucket: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode(), e);
            throw e;
        }
    }
```

- Per i dettagli sull'API, consulta la sezione AWS SDK for Java 2.x API CreateBucketReference.

`CreateMultipartUpload`

Il seguente esempio di codice mostra come utilizzare`CreateMultipartUpload`.

SDK per Java 2.x

> **Note**
>
> C'è 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 un caricamento in più parti in un bucket di directory.

```
import com.example.s3.util.S3DirectoryBucketUtils;
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.model.CreateMultipartUploadRequest;
import software.amazon.awssdk.services.s3.model.CreateMultipartUploadResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;

    /**
     * This method creates a multipart upload request that generates a unique upload
     * ID used to track
     * all the upload parts.
     *
     * @param s3Client   The S3 client used to interact with S3
     * @param bucketName The name of the directory bucket
     * @param objectKey  The key (name) of the object to be uploaded
     * @return The upload ID used to track the multipart upload
     */
    public static String createDirectoryBucketMultipartUpload(S3Client s3Client,
 String bucketName, String objectKey) {
        logger.info("Creating multipart upload for object: {} in bucket: {}",
 objectKey, bucketName);

        try {
            // Create a CreateMultipartUploadRequest
            CreateMultipartUploadRequest createMultipartUploadRequest =
 CreateMultipartUploadRequest.builder()
                    .bucket(bucketName)
                    .key(objectKey)
                    .build();

            // Initiate the multipart upload
            CreateMultipartUploadResponse response =
 s3Client.createMultipartUpload(createMultipartUploadRequest);
            String uploadId = response.uploadId();
            logger.info("Multipart upload initiated. Upload ID: {}", uploadId);
            return uploadId;
```

```
        } catch (S3Exception e) {
            logger.error("Failed to create multipart upload: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode(), e);
            throw e;
        }
    }
```

- Per i dettagli sull'API, consulta la sezione AWS SDK for Java 2.x API CreateMultipartUploadReference.

`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.

Elimina un bucket di directory S3.

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

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;

    /**
     * Deletes the specified S3 directory bucket.
     *
```

```
     * @param s3Client   The S3 client used to interact with S3
     * @param bucketName The name of the directory bucket to delete
     */
    public static void deleteDirectoryBucket(S3Client s3Client, String bucketName) {
        logger.info("Deleting bucket: {}", bucketName);

        try {
            // Create a DeleteBucketRequest
            DeleteBucketRequest deleteBucketRequest = DeleteBucketRequest.builder()
                    .bucket(bucketName)
                    .build();

            // Delete the bucket
            s3Client.deleteBucket(deleteBucketRequest);
            logger.info("Successfully deleted bucket: {}", bucketName);

        } catch (S3Exception e) {
            logger.error("Failed to delete bucket: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode(), e);
            throw e;
        }
    }
```

- Per i dettagli sull'API, consulta la sezione AWS SDK for Java 2.x API DeleteBucketReference.

`DeleteBucketEncryption`

Il seguente esempio di codice mostra come utilizzare`DeleteBucketEncryption`.

SDK per Java 2.x

> **Note**
>
> C'è 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 la configurazione di crittografia per un bucket di directory.

```
import software.amazon.awssdk.regions.Region;
```

```
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.DeleteBucketEncryptionRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;

    /**
     * Deletes the encryption configuration from an S3 bucket.
     *
     * @param s3Client   The S3 client used to interact with S3
     * @param bucketName The name of the directory bucket
     */
    public static void deleteDirectoryBucketEncryption(S3Client s3Client, String
 bucketName) {
        DeleteBucketEncryptionRequest deleteRequest =
 DeleteBucketEncryptionRequest.builder()
                .bucket(bucketName)
                .build();

        try {
            s3Client.deleteBucketEncryption(deleteRequest);
            logger.info("Bucket encryption deleted for bucket: {}", bucketName);
        } catch (S3Exception e) {
            logger.error("Failed to delete bucket encryption: {} - Error code: {}",
 e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode(), e);
            throw e;
        }
    }
```

- Per i dettagli sull'API, consulta la sezione DeleteBucketEncryption AWS SDK for Java 2.xAPI 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.

Elimina una policy relativa ai bucket per un bucket di directory.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.DeleteBucketPolicyRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getAwsAccountId;
import static com.example.s3.util.S3DirectoryBucketUtils.putDirectoryBucketPolicy;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;


    /**
     * Deletes the bucket policy for the specified S3 directory bucket.
     *
     * @param s3Client   The S3 client used to interact with S3
     * @param bucketName The name of the directory bucket
     */
    public static void deleteDirectoryBucketPolicy(S3Client s3Client, String
 bucketName) {
        logger.info("Deleting policy for bucket: {}", bucketName);

        try {
            // Create a DeleteBucketPolicyRequest
            DeleteBucketPolicyRequest deletePolicyReq =
 DeleteBucketPolicyRequest.builder()
                    .bucket(bucketName)
                    .build();
```

```
            // Delete the bucket policy
            s3Client.deleteBucketPolicy(deletePolicyReq);
            logger.info("Successfully deleted bucket policy");

        } catch (S3Exception e) {
            logger.error("Failed to delete bucket policy: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode(), e);
            throw e;
        }
    }
```

- Per i dettagli sull'API, consulta la sezione AWS SDK for Java 2.x API DeleteBucketPolicyReference.

`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.

Elimina un oggetto in un bucket di directory.

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

import java.nio.file.Path;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
```

```
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;
import static com.example.s3.util.S3DirectoryBucketUtils.putDirectoryBucketObject;




    /**
     * Deletes an object from the specified S3 directory bucket.
     *
     * @param s3Client   The S3 client used to interact with S3
     * @param bucketName The name of the directory bucket
     * @param objectKey  The key (name) of the object to be deleted
     */
    public static void deleteDirectoryBucketObject(S3Client s3Client, String
 bucketName, String objectKey) {
        logger.info("Deleting object: {} from bucket: {}", objectKey, bucketName);

        try {
            // Create a DeleteObjectRequest
            DeleteObjectRequest deleteObjectRequest = DeleteObjectRequest.builder()
                    .bucket(bucketName)
                    .key(objectKey)
                    .build();

            // Delete the object
            s3Client.deleteObject(deleteObjectRequest);
            logger.info("Object {} has been deleted", objectKey);

        } catch (S3Exception e) {
            logger.error("Failed to delete object: {} - Error code: {}",
 e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode(), e);
            throw e;
        }
    }
```

- Per i dettagli sull'API, consulta la sezione DeleteObject AWS SDK for Java 2.xAPI 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.

Elimina più oggetti in un bucket di directory.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.Delete;
import software.amazon.awssdk.services.s3.model.DeleteObjectsRequest;
import software.amazon.awssdk.services.s3.model.DeleteObjectsResponse;
import software.amazon.awssdk.services.s3.model.ObjectIdentifier;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.net.URISyntaxException;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.List;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;
import static com.example.s3.util.S3DirectoryBucketUtils.putDirectoryBucketObject;


    /**
     * Deletes multiple objects from the specified S3 directory bucket.
     *
     * @param s3Client   The S3 client used to interact with S3
     * @param bucketName The name of the directory bucket
     * @param objectKeys The list of keys (names) of the objects to be deleted
     */
    public static void deleteDirectoryBucketObjects(S3Client s3Client, String
 bucketName, List<String> objectKeys) {
```

```
        logger.info("Deleting objects from bucket: {}", bucketName);

        try {
            // Create a list of ObjectIdentifier.
            List<ObjectIdentifier> identifiers = objectKeys.stream()
                    .map(key -> ObjectIdentifier.builder().key(key).build())
                    .toList();

            Delete delete = Delete.builder()
                    .objects(identifiers)
                    .build();

            DeleteObjectsRequest deleteObjectsRequest =
DeleteObjectsRequest.builder()
                    .bucket(bucketName)
                    .delete(delete)
                    .build();

            DeleteObjectsResponse deleteObjectsResponse =
s3Client.deleteObjects(deleteObjectsRequest);
            deleteObjectsResponse.deleted().forEach(deleted -> logger.info("Deleted
object: {}", deleted.key()));

        } catch (S3Exception e) {
            logger.error("Failed to delete objects: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode(), e);
            throw e;
        }
    }
```

- Per i dettagli sull'API, consulta la sezione DeleteObjects AWS SDK for Java 2.xAPI Reference.

`GetBucketEncryption`

Il seguente esempio di codice mostra come utilizzare`GetBucketEncryption`.

SDK per Java 2.x

> **Note**
>
> C'è 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 la configurazione di crittografia di un bucket di directory.

```
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.model.GetBucketEncryptionRequest;
import software.amazon.awssdk.services.s3.model.GetBucketEncryptionResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.ServerSideEncryptionRule;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;

    /**
     * Retrieves the encryption configuration for an S3 directory bucket.
     *
     * @param s3Client   The S3 client used to interact with S3
     * @param bucketName The name of the directory bucket
     * @return The type of server-side encryption applied to the bucket (e.g.,
     *         AES256, aws:kms)
     */
    public static String getDirectoryBucketEncryption(S3Client s3Client, String
 bucketName) {
        try {
            // Create a GetBucketEncryptionRequest
            GetBucketEncryptionRequest getRequest =
 GetBucketEncryptionRequest.builder()
                    .bucket(bucketName)
                    .build();

            // Retrieve the bucket encryption configuration
```

```
            GetBucketEncryptionResponse response =
s3Client.getBucketEncryption(getRequest);
            ServerSideEncryptionRule rule =
response.serverSideEncryptionConfiguration().rules().get(0);

            String encryptionType =
rule.applyServerSideEncryptionByDefault().sseAlgorithmAsString();
            logger.info("Bucket encryption algorithm: {}", encryptionType);
            logger.info("KMS Customer Managed Key ID: {}",
rule.applyServerSideEncryptionByDefault().kmsMasterKeyID());
            logger.info("Bucket Key Enabled: {}", rule.bucketKeyEnabled());

            return encryptionType;
        } catch (S3Exception e) {
            logger.error("Failed to get bucket encryption: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode(), e);
            throw e;
        }
    }
```

- Per i dettagli sull'API, consulta la sezione GetBucketEncryption AWS SDK for Java 2.xAPI 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.

Ottieni la politica di un bucket di directory.

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

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getAwsAccountId;
import static com.example.s3.util.S3DirectoryBucketUtils.putDirectoryBucketPolicy;

    /**
     * Retrieves the bucket policy for the specified S3 directory bucket.
     *
     * @param s3Client   The S3 client used to interact with S3
     * @param bucketName The name of the directory bucket
     * @return The bucket policy text
     */
    public static String getDirectoryBucketPolicy(S3Client s3Client, String
 bucketName) {
        logger.info("Getting policy for bucket: {}", bucketName);

        try {
            // Create a GetBucketPolicyRequest
            GetBucketPolicyRequest policyReq = GetBucketPolicyRequest.builder()
                    .bucket(bucketName)
                    .build();

            // Retrieve the bucket policy
            GetBucketPolicyResponse response = s3Client.getBucketPolicy(policyReq);

            // Print and return the policy text
            String policyText = response.policy();
            logger.info("Bucket policy: {}", policyText);
            return policyText;

        } catch (S3Exception e) {
            logger.error("Failed to get bucket policy: {} - Error code: {}",
 e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode(), e);
            throw e;
        }
    }
```

- Per i dettagli sull'API, consulta la sezione GetBucketPolicy AWS SDK for Java 2.xAPI 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.

Ottieni un oggetto da un bucket di directory.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
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.GetObjectResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.nio.charset.StandardCharsets;
import java.nio.file.Path;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static
 com.example.s3.util.S3DirectoryBucketUtils.deleteAllObjectsInDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;
import static com.example.s3.util.S3DirectoryBucketUtils.putDirectoryBucketObject;

    /**
```

```
     * Retrieves an object from the specified S3 directory bucket.
     *
     * @param s3Client   The S3 client used to interact with S3
     * @param bucketName The name of the directory bucket
     * @param objectKey  The key (name) of the object to be retrieved
     * @return The retrieved object as a ResponseInputStream
     */
    public static boolean getDirectoryBucketObject(S3Client s3Client, String
bucketName, String objectKey) {
        logger.info("Retrieving object: {} from bucket: {}", objectKey, bucketName);

        try {
            // Create a GetObjectRequest
            GetObjectRequest objectRequest = GetObjectRequest.builder()
                    .key(objectKey)
                    .bucket(bucketName)
                    .build();

            // Retrieve the object as bytes
            ResponseBytes<GetObjectResponse> objectBytes =
s3Client.getObjectAsBytes(objectRequest);
            byte[] data = objectBytes.asByteArray();

            // Print object contents to console
            String objectContent = new String(data, StandardCharsets.UTF_8);
            logger.info("Object contents: \n{}", objectContent);

            return true;

        } catch (S3Exception e) {
            logger.error("Failed to retrieve object: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode(), e);
            return false;
        }
    }
```

- Per i dettagli sull'API, consulta la sezione GetObject AWS SDK for Java 2.xAPI Reference.

`GetObjectAttributes`

Il seguente esempio di codice mostra come utilizzare`GetObjectAttributes`.

SDK per Java 2.x

> **Note**
>
> C'è 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 gli attributi di un oggetto da un bucket di directory.

```
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.model.GetObjectAttributesRequest;
import software.amazon.awssdk.services.s3.model.GetObjectAttributesResponse;
import software.amazon.awssdk.services.s3.model.ObjectAttributes;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.nio.file.Path;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static
 com.example.s3.util.S3DirectoryBucketUtils.deleteAllObjectsInDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;
import static com.example.s3.util.S3DirectoryBucketUtils.putDirectoryBucketObject;

    /**
     * Retrieves attributes for an object in the specified S3 directory bucket.
     *
     * @param s3Client   The S3 client used to interact with S3
     * @param bucketName The name of the directory bucket
     * @param objectKey  The key (name) of the object to retrieve attributes for
     * @return True if the object attributes are successfully retrieved, false
     *         otherwise
     */
    public static boolean getDirectoryBucketObjectAttributes(S3Client s3Client,
 String bucketName, String objectKey) {
        logger.info("Retrieving attributes for object: {} from bucket: {}",
 objectKey, bucketName);
```

```
        try {
            // Create a GetObjectAttributesRequest
            GetObjectAttributesRequest getObjectAttributesRequest =
GetObjectAttributesRequest.builder()
                    .bucket(bucketName)
                    .key(objectKey)
                    .objectAttributes(ObjectAttributes.E_TAG,
ObjectAttributes.STORAGE_CLASS,
                            ObjectAttributes.OBJECT_SIZE)
                    .build();

            // Retrieve the object attributes
            GetObjectAttributesResponse response =
s3Client.getObjectAttributes(getObjectAttributesRequest);
            logger.info("Attributes for object {}:", objectKey);
            logger.info("ETag: {}", response.eTag());
            logger.info("Storage Class: {}", response.storageClass());
            logger.info("Object Size: {}", response.objectSize());
            return true;

        } catch (S3Exception e) {
            logger.error("Failed to retrieve object attributes: {} - Error code:
{}",
                    e.awsErrorDetails().errorMessage(),
e.awsErrorDetails().errorCode(), e);
            return false;
        }
    }
```

- Per i dettagli sull'API, consulta la sezione GetObjectAttributes AWS SDK for Java 2.xAPI Reference.

`HeadBucket`

Il seguente esempio di codice mostra come utilizzare`HeadBucket`.

SDK per Java 2.x

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

Verifica se il bucket di directory S3 specificato esiste ed è accessibile.

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

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;

    /**
     * Checks if the specified S3 directory bucket exists and is accessible.
     *
     * @param s3Client   The S3 client used to interact with S3
     * @param bucketName The name of the directory bucket to check
     * @return True if the bucket exists and is accessible, false otherwise
     */
    public static boolean headDirectoryBucket(S3Client s3Client, String bucketName)
 {
        logger.info("Checking if bucket exists: {}", bucketName);

        try {
            // Create a HeadBucketRequest
            HeadBucketRequest headBucketRequest = HeadBucketRequest.builder()
                    .bucket(bucketName)
                    .build();
            // If the bucket doesn't exist, the following statement throws
 NoSuchBucketException,
            // which is a subclass of S3Exception.
            s3Client.headBucket(headBucketRequest);
            logger.info("Amazon S3 directory bucket: \"{}\" found.", bucketName);
```

```
            return true;

        } catch (S3Exception e) {
            logger.error("Failed to access bucket: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode(), e);
            throw e;
        }
    }
```

- Per i dettagli sull'API, consulta la sezione AWS SDK for Java 2.x API HeadBucketReference.

`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.

Ottieni i metadati di un oggetto in un bucket di directory.

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

import java.nio.file.Path;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static
 com.example.s3.util.S3DirectoryBucketUtils.deleteAllObjectsInDirectoryBucket;
```

```
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;
import static com.example.s3.util.S3DirectoryBucketUtils.putDirectoryBucketObject;

    /**
     * Retrieves metadata for an object in the specified S3 directory bucket.
     *
     * @param s3Client   The S3 client used to interact with S3
     * @param bucketName The name of the directory bucket
     * @param objectKey  The key (name) of the object to retrieve metadata for
     * @return True if the object exists, false otherwise
     */
    public static boolean headDirectoryBucketObject(S3Client s3Client, String
 bucketName, String objectKey) {
        logger.info("Retrieving metadata for object: {} from bucket: {}", objectKey,
 bucketName);

        try {
            // Create a HeadObjectRequest
            HeadObjectRequest headObjectRequest = HeadObjectRequest.builder()
                    .bucket(bucketName)
                    .key(objectKey)
                    .build();

            // Retrieve the object metadata
            HeadObjectResponse response = s3Client.headObject(headObjectRequest);
            logger.info("Amazon S3 object: \"{}\" found in bucket: \"{}\" with ETag:
 \"{}\"", objectKey, bucketName,
                    response.eTag());
            logger.info("Content-Type: {}", response.contentType());
            logger.info("Content-Length: {}", response.contentLength());
            logger.info("Last Modified: {}", response.lastModified());
            return true;

        } catch (S3Exception e) {
            logger.error("Failed to retrieve object metadata: {} - Error code: {}",
 e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode(), e);
            return false;
        }
    }
```

- Per i dettagli sull'API, consulta AWS SDK for Java 2.x API HeadObjectReference.

`ListDirectoryBuckets`

Il seguente esempio di codice mostra come utilizzare`ListDirectoryBuckets`.

SDK per Java 2.x

> **Note**
>
> C'è 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 bucket di directory.

```
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.model.Bucket;
import software.amazon.awssdk.services.s3.model.ListDirectoryBucketsRequest;
import software.amazon.awssdk.services.s3.model.ListDirectoryBucketsResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.util.List;
import java.util.UUID;
import java.util.stream.Collectors;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;

     /**
     * Lists all S3 directory buckets and no general purpose buckets.
     *
     * @param s3Client The S3 client used to interact with S3
     * @return A list of bucket names
     */
    public static List<String> listDirectoryBuckets(S3Client s3Client) {
        logger.info("Listing all directory buckets");

        try {
            // Create a ListBucketsRequest
```

```
            ListDirectoryBucketsRequest listDirectoryBucketsRequest =
ListDirectoryBucketsRequest.builder().build();

            // Retrieve the list of buckets
            ListDirectoryBucketsResponse response =
s3Client.listDirectoryBuckets(listDirectoryBucketsRequest);

            // Extract bucket names
            List<String> bucketNames = response.buckets().stream()
                    .map(Bucket::name)
                    .collect(Collectors.toList());

            return bucketNames;
        } catch (S3Exception e) {
            logger.error("Failed to list buckets: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode());
            throw e;
        }
    }
```

- Per i dettagli sull'API, consulta la sezione ListDirectoryBuckets AWS SDK for Java 2.xAPI 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.

Elenca i caricamenti in più parti in un bucket di directory.

```
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.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.io.IOException;
import java.nio.file.Path;
import java.util.List;

import static
 com.example.s3.util.S3DirectoryBucketUtils.abortDirectoryBucketMultipartUploads;
import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static
 com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucketMultipartUpload;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;
import static
 com.example.s3.util.S3DirectoryBucketUtils.multipartUploadForDirectoryBucket;


   /**
    * Lists multipart uploads for the specified S3 directory bucket.
    *
    * @param s3Client   The S3 client used to interact with S3
    * @param bucketName The name of the directory bucket
    * @return A list of MultipartUpload objects representing the multipart uploads
    */
   public static List<MultipartUpload> listDirectoryBucketMultipartUploads(S3Client
 s3Client, String bucketName) {
       logger.info("Listing in-progress multipart uploads for bucket: {}",
 bucketName);

       try {
           // Create a ListMultipartUploadsRequest
           ListMultipartUploadsRequest listMultipartUploadsRequest =
 ListMultipartUploadsRequest.builder()
                   .bucket(bucketName)
                   .build();

           // List the multipart uploads
```

```
            ListMultipartUploadsResponse response =
s3Client.listMultipartUploads(listMultipartUploadsRequest);
            List<MultipartUpload> uploads = response.uploads();
            for (MultipartUpload upload : uploads) {
                logger.info("In-progress multipart upload: Upload ID: {}, Key: {},
Initiated: {}", upload.uploadId(),
                        upload.key(), upload.initiated());
            }
            return uploads;

        } catch (S3Exception e) {
            logger.error("Failed to list multipart uploads: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode());
            return List.of(); // Return an empty list if an exception is thrown
        }
    }
```

- Per i dettagli sull'API, consulta la sezione API ListMultipartUploadsReference AWS SDK for Java 2.x .

`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.

Elenca gli oggetti in un bucket di directory.

```
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.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;

import java.nio.file.Path;
import java.util.List;
import java.util.stream.Collectors;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static
 com.example.s3.util.S3DirectoryBucketUtils.deleteAllObjectsInDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;
import static com.example.s3.util.S3DirectoryBucketUtils.putDirectoryBucketObject;

    /**
     * Lists objects in the specified S3 directory bucket.
     *
     * @param s3Client   The S3 client used to interact with S3
     * @param bucketName The name of the directory bucket
     * @return A list of object keys in the bucket
     */
    public static List<String> listDirectoryBucketObjectsV2(S3Client s3Client,
 String bucketName) {
        logger.info("Listing objects in bucket: {}", bucketName);

        try {
            // Create a ListObjectsV2Request
            ListObjectsV2Request listObjectsV2Request =
 ListObjectsV2Request.builder()
                    .bucket(bucketName)
                    .build();

            // Retrieve the list of objects
            ListObjectsV2Response response =
 s3Client.listObjectsV2(listObjectsV2Request);

            // Extract and return the object keys
            return response.contents().stream()
                    .map(S3Object::key)
                    .collect(Collectors.toList());

        } catch (S3Exception e) {
```

```
            logger.error("Failed to list objects: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode());
            throw e;
        }
    }
```

- Per i dettagli sull'API, consulta ListObjectsV2 in AWS SDK for Java 2.x API Reference.

`ListParts`

Il seguente esempio di codice mostra come utilizzare`ListParts`.

SDK per Java 2.x

> **Note**
>
> C'è 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 le parti di un caricamento in più parti in un bucket di directory.

```
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.model.ListPartsRequest;
import software.amazon.awssdk.services.s3.model.ListPartsResponse;
import software.amazon.awssdk.services.s3.model.Part;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.io.IOException;
import java.nio.file.Path;
import java.util.List;

import static
 com.example.s3.util.S3DirectoryBucketUtils.abortDirectoryBucketMultipartUploads;
import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
```

```
import static
 com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucketMultipartUpload;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;
import static
 com.example.s3.util.S3DirectoryBucketUtils.multipartUploadForDirectoryBucket;

   /**
    * Lists the parts of a multipart upload for the specified S3 directory bucket.
    *
    * @param s3Client   The S3 client used to interact with S3
    * @param bucketName The name of the directory bucket
    * @param objectKey  The key (name) of the object being uploaded
    * @param uploadId   The upload ID used to track the multipart upload
    * @return A list of Part representing the parts of the multipart upload
    */
   public static List<Part> listDirectoryBucketMultipartUploadParts(S3Client
 s3Client, String bucketName,
           String objectKey, String uploadId) {
       logger.info("Listing parts for object: {} in bucket: {}", objectKey,
 bucketName);

       try {
           // Create a ListPartsRequest
           ListPartsRequest listPartsRequest = ListPartsRequest.builder()
                   .bucket(bucketName)
                   .uploadId(uploadId)
                   .key(objectKey)
                   .build();

           // List the parts of the multipart upload
           ListPartsResponse response = s3Client.listParts(listPartsRequest);
           List<Part> parts = response.parts();
           for (Part part : parts) {
               logger.info("Uploaded part: Part number = \"{}\", etag = {}",
 part.partNumber(), part.eTag());
           }
           return parts;

       } catch (S3Exception e) {
           logger.error("Failed to list parts: {} - Error code: {}",
 e.awsErrorDetails().errorMessage(),
                   e.awsErrorDetails().errorCode());
```

```
            return List.of(); // Return an empty list if an exception is thrown
        }
    }
```

- Per i dettagli sull'API, consulta la sezione AWS SDK for Java 2.x API ListPartsReference.

`PutBucketEncryption`

Il seguente esempio di codice mostra come utilizzare`PutBucketEncryption`.

SDK per Java 2.x

> **Note**
>
> C'è 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 crittografia dei bucket su un bucket di directory.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.kms.KmsClient;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutBucketEncryptionRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.ServerSideEncryption;
import software.amazon.awssdk.services.s3.model.ServerSideEncryptionByDefault;
import software.amazon.awssdk.services.s3.model.ServerSideEncryptionConfiguration;
import software.amazon.awssdk.services.s3.model.ServerSideEncryptionRule;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createKmsClient;
import static com.example.s3.util.S3DirectoryBucketUtils.createKmsKey;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.scheduleKeyDeletion;

    /**
     * Sets the default encryption configuration for an S3 bucket as SSE-KMS.
```

```
     *
     * @param s3Client   The S3 client used to interact with S3
     * @param bucketName The name of the directory bucket
     * @param kmsKeyId   The ID of the customer-managed KMS key
     */
    public static void putDirectoryBucketEncryption(S3Client s3Client, String
bucketName, String kmsKeyId) {
        // Define the default encryption configuration to use SSE-KMS. For directory
        // buckets, AWS managed KMS keys aren't supported. Only customer-managed
keys
        // are supported.
        ServerSideEncryptionByDefault encryptionByDefault =
ServerSideEncryptionByDefault.builder()
                .sseAlgorithm(ServerSideEncryption.AWS_KMS)
                .kmsMasterKeyID(kmsKeyId)
                .build();

        // Create a server-side encryption rule to apply the default encryption
        // configuration. For directory buckets, the bucketKeyEnabled field is
enforced
        // to be true.
        ServerSideEncryptionRule rule = ServerSideEncryptionRule.builder()
                .bucketKeyEnabled(true)
                .applyServerSideEncryptionByDefault(encryptionByDefault)
                .build();

        // Create the server-side encryption configuration for the bucket
        ServerSideEncryptionConfiguration encryptionConfiguration =
ServerSideEncryptionConfiguration.builder()
                .rules(rule)
                .build();

        // Create the PutBucketEncryption request
        PutBucketEncryptionRequest putRequest = PutBucketEncryptionRequest.builder()
                .bucket(bucketName)
                .serverSideEncryptionConfiguration(encryptionConfiguration)
                .build();

        // Set the bucket encryption
        try {
            s3Client.putBucketEncryption(putRequest);
            logger.info("SSE-KMS Bucket encryption configuration set for the
directory bucket: {}", bucketName);
        } catch (S3Exception e) {
```

```
            logger.error("Failed to set bucket encryption: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode());
            throw e;
        }
    }
```

- Per i dettagli sull'API, consulta la sezione AWS SDK for Java 2.x API PutBucketEncryptionReference.

`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.

Applica una policy sui bucket a un bucket di directory.

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

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getAwsAccountId;

    /**
     * Sets the following bucket policy for the specified S3 directory bucket.
     *<pre>
     * {
```

```
     *     "Version": "2012-10-17",
     *     "Statement": [
     *         {
     *             "Sid": "AdminPolicy",
     *             "Effect": "Allow",
     *             "Principal": {
     *                 "AWS": "arn:aws:iam::<ACCOUNT_ID>:root"
     *             },
     *             "Action": "s3express:*",
     *             "Resource": "arn:aws:s3express:us-west-2:<ACCOUNT_ID>:bucket/
<DIR_BUCKET_NAME>
     *         }
     *     ]
     * }
     * </pre>
     * This policy grants all S3 directory bucket actions to identities in the same
 account as the bucket.
     *
     * @param s3Client   The S3 client used to interact with S3
     * @param bucketName The name of the directory bucket
     * @param policyText The policy text to be applied
     */
    public static void putDirectoryBucketPolicy(S3Client s3Client, String
 bucketName, String policyText) {
        logger.info("Setting policy on bucket: {}", bucketName);
        logger.info("Policy: {}", policyText);

        try {
            PutBucketPolicyRequest policyReq = PutBucketPolicyRequest.builder()
                    .bucket(bucketName)
                    .policy(policyText)
                    .build();

            s3Client.putBucketPolicy(policyReq);
            logger.info("Bucket policy set successfully!");

        } catch (S3Exception e) {
            logger.error("Failed to set bucket policy: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode(), e);
            throw e;
        }
    }
```

- Per i dettagli sull'API, consulta la sezione AWS SDK for Java 2.x API PutBucketPolicyReference.

`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.

Inserisci un oggetto in un bucket di directory.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.awscore.exception.AwsErrorDetails;
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.UncheckedIOException;
import java.nio.file.Path;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static
 com.example.s3.util.S3DirectoryBucketUtils.deleteAllObjectsInDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;

    /**
     * Puts an object into the specified S3 directory bucket.
     *
```

```
     * @param s3Client   The S3 client used to interact with S3
     * @param bucketName The name of the directory bucket
     * @param objectKey  The key (name) of the object to be placed in the bucket
     * @param filePath   The path of the file to be uploaded
     */
    public static void putDirectoryBucketObject(S3Client s3Client, String
bucketName, String objectKey, Path filePath) {
        logger.info("Putting object: {} into bucket: {}", objectKey, bucketName);

        try {
            // Create a PutObjectRequest
            PutObjectRequest putObj = PutObjectRequest.builder()
                    .bucket(bucketName)
                    .key(objectKey)
                    .build();

            // Upload the object
            s3Client.putObject(putObj, filePath);
            logger.info("Successfully placed {} into bucket {}", objectKey,
bucketName);

        } catch (UncheckedIOException e) {
            throw S3Exception.builder().message("Failed to read the file: " +
e.getMessage()).cause(e)
                    .awsErrorDetails(AwsErrorDetails.builder()
                            .errorCode("ClientSideException:FailedToReadFile")
                            .errorMessage(e.getMessage())
                            .build())
                    .build();
        } catch (S3Exception e) {
            logger.error("Failed to put object: {}", e.getMessage(), e);
            throw e;
        }
    }
```

- Per i dettagli sull'API, consulta la sezione PutObject AWS SDK for Java 2.xAPI Reference.

`UploadPart`

Il seguente esempio di codice mostra come utilizzare`UploadPart`.

SDK per Java 2.x

> **Note**
>
> C'è 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 parte di un caricamento in più parti per un bucket di directory.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
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.CompletedPart;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.UploadPartRequest;
import software.amazon.awssdk.services.s3.model.UploadPartResponse;

import java.io.IOException;
import java.io.RandomAccessFile;
import java.nio.ByteBuffer;
import java.nio.file.Path;
import java.util.ArrayList;
import java.util.List;

import static
 com.example.s3.util.S3DirectoryBucketUtils.abortDirectoryBucketMultipartUploads;
import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static
 com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucketMultipartUpload;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static
 com.example.s3.util.S3DirectoryBucketUtils.deleteAllObjectsInDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;

    /**
     * This method creates part requests and uploads individual parts to S3.
     * While it uses the UploadPart API to upload a single part, it does so
     * sequentially to handle multiple parts of a file, returning all the completed
```

```
     * parts.
     *
     * @param s3Client   The S3 client used to interact with S3
     * @param bucketName The name of the directory bucket
     * @param objectKey  The key (name) of the object to be uploaded
     * @param uploadId   The upload ID used to track the multipart upload
     * @param filePath   The path to the file to be uploaded
     * @return A list of uploaded parts
     * @throws IOException if an I/O error occurs
     */
    public static List<CompletedPart> multipartUploadForDirectoryBucket(S3Client
s3Client, String bucketName,
            String objectKey, String uploadId, Path filePath) throws IOException {
        logger.info("Uploading parts for object: {} in bucket: {}", objectKey,
bucketName);

        int partNumber = 1;
        List<CompletedPart> uploadedParts = new ArrayList<>();
        ByteBuffer bb = ByteBuffer.allocate(1024 * 1024 * 5); // 5 MB byte buffer

        // Read the local file, break down into chunks and process
        try (RandomAccessFile file = new RandomAccessFile(filePath.toFile(), "r")) {
            long fileSize = file.length();
            int position = 0;

            // Sequentially upload parts of the file
            while (position < fileSize) {
                file.seek(position);
                int read = file.getChannel().read(bb);

                bb.flip(); // Swap position and limit before reading from the buffer
                UploadPartRequest uploadPartRequest = UploadPartRequest.builder()
                        .bucket(bucketName)
                        .key(objectKey)
                        .uploadId(uploadId)
                        .partNumber(partNumber)
                        .build();

                UploadPartResponse partResponse = s3Client.uploadPart(
                        uploadPartRequest,
                        RequestBody.fromByteBuffer(bb));

                // Build the uploaded part
                CompletedPart uploadedPart = CompletedPart.builder()
```

```
                        .partNumber(partNumber)
                        .eTag(partResponse.eTag())
                        .build();

                // Add the uploaded part to the list
                uploadedParts.add(uploadedPart);

                // Log to indicate the part upload is done
                logger.info("Uploaded part number: {} with ETag: {}", partNumber,
partResponse.eTag());

                bb.clear();
                position += read;
                partNumber++;
            }
        } catch (S3Exception e) {
            logger.error("Failed to list parts: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode());
            throw e;
        }
        return uploadedParts;
    }
```

- Per i dettagli sull'API, consulta la sezione AWS SDK for Java 2.x API UploadPartReference.

`UploadPartCopy`

Il seguente esempio di codice mostra come utilizzare`UploadPartCopy`.

SDK per Java 2.x

> **Note**
>
> C'è 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 parti di copia in base alla dimensione dell'oggetto sorgente e copia le singole parti in un bucket di directory.

```
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.model.CompletedPart;
import software.amazon.awssdk.services.s3.model.HeadObjectRequest;
import software.amazon.awssdk.services.s3.model.HeadObjectResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.UploadPartCopyRequest;
import software.amazon.awssdk.services.s3.model.UploadPartCopyResponse;

import java.io.IOException;
import java.nio.file.Path;
import java.util.ArrayList;
import java.util.List;

import static
 com.example.s3.util.S3DirectoryBucketUtils.abortDirectoryBucketMultipartUploads;
import static
 com.example.s3.util.S3DirectoryBucketUtils.completeDirectoryBucketMultipartUpload;
import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static
 com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucketMultipartUpload;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static
 com.example.s3.util.S3DirectoryBucketUtils.deleteAllObjectsInDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;
import static
 com.example.s3.util.S3DirectoryBucketUtils.multipartUploadForDirectoryBucket;

    /**
     * Creates copy parts based on source object size and copies over individual
     * parts.
     *
     * @param s3Client          The S3 client used to interact with S3
     * @param sourceBucket      The name of the source bucket
     * @param sourceKey         The key (name) of the source object
     * @param destinationBucket The name of the destination bucket
     * @param destinationKey    The key (name) of the destination object
     * @param uploadId          The upload ID used to track the multipart upload
     * @return A list of completed parts
```

```
    */
   public static List<CompletedPart> multipartUploadCopyForDirectoryBucket(S3Client
s3Client, String sourceBucket,
           String sourceKey, String destinationBucket, String destinationKey,
String uploadId) {
       // Get the object size to track the end of the copy operation
       HeadObjectRequest headObjectRequest = HeadObjectRequest.builder()
               .bucket(sourceBucket)
               .key(sourceKey)
               .build();
       HeadObjectResponse headObjectResponse =
s3Client.headObject(headObjectRequest);
       long objectSize = headObjectResponse.contentLength();

       logger.info("Source Object size: {}", objectSize);

       // Copy the object using 20 MB parts
       long partSize = 20 * 1024 * 1024; // 20 MB
       long bytePosition = 0;
       int partNum = 1;
       List<CompletedPart> uploadedParts = new ArrayList<>();

       while (bytePosition < objectSize) {
           long lastByte = Math.min(bytePosition + partSize - 1, objectSize - 1);
           logger.info("Part Number: {}, Byte Position: {}, Last Byte: {}",
partNum, bytePosition, lastByte);

           try {
               UploadPartCopyRequest uploadPartCopyRequest =
UploadPartCopyRequest.builder()
                       .sourceBucket(sourceBucket)
                       .sourceKey(sourceKey)
                       .destinationBucket(destinationBucket)
                       .destinationKey(destinationKey)
                       .uploadId(uploadId)
                       .copySourceRange("bytes=" + bytePosition + "-" + lastByte)
                       .partNumber(partNum)
                       .build();
               UploadPartCopyResponse uploadPartCopyResponse =
s3Client.uploadPartCopy(uploadPartCopyRequest);

               CompletedPart part = CompletedPart.builder()
                       .partNumber(partNum)
                       .eTag(uploadPartCopyResponse.copyPartResult().eTag())
```

```
                        .build();
                uploadedParts.add(part);

                bytePosition += partSize;
                partNum++;
            } catch (S3Exception e) {
                logger.error("Failed to copy part number {}: {} - Error code: {}",
partNum,
                        e.awsErrorDetails().errorMessage(),
e.awsErrorDetails().errorCode());
                throw e;
            }
        }

        return uploadedParts;
    }
```

- Per i dettagli sull'API, consulta la sezione UploadPartCopy AWS SDK for Java 2.xAPI Reference.

Scenari

Crea un URL predefinito per ottenere un oggetto

Il seguente esempio di codice mostra come creare un URL predefinito per i bucket di directory S3 e ottenere un oggetto.

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.

Genera un URL GET predefinito per accedere a un oggetto in un bucket di directory S3.

```
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.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 java.nio.file.Path;
import java.time.Duration;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Presigner;
import static
 com.example.s3.util.S3DirectoryBucketUtils.deleteAllObjectsInDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;
import static com.example.s3.util.S3DirectoryBucketUtils.putDirectoryBucketObject;


   /**
    * Generates a presigned URL for accessing an object in the specified S3
    * directory bucket.
    *
    * @param s3Presigner The S3 presigner client used to generate the presigned URL
    * @param bucketName  The name of the directory bucket
    * @param objectKey   The key (name) of the object to access
    * @return A presigned URL for accessing the specified object
    */
   public static String generatePresignedGetURLForDirectoryBucket(S3Presigner
 s3Presigner, String bucketName,
           String objectKey) {
       logger.info("Generating presigned URL for object: {} in bucket: {}",
 objectKey, bucketName);

       try {
           // Create a GetObjectRequest
           GetObjectRequest getObjectRequest = GetObjectRequest.builder()
                   .bucket(bucketName)
                   .key(objectKey)
                   .build();

           // Create a GetObjectPresignRequest
```

```
            GetObjectPresignRequest getObjectPresignRequest =
GetObjectPresignRequest.builder()
                    .signatureDuration(Duration.ofMinutes(10)) // Presigned URL
valid for 10 minutes
                    .getObjectRequest(getObjectRequest)
                    .build();

            // Generate the presigned URL
            PresignedGetObjectRequest presignedGetObjectRequest =
s3Presigner.presignGetObject(getObjectPresignRequest);

            // Get the presigned URL
            String presignedURL = presignedGetObjectRequest.url().toString();
            logger.info("Presigned URL: {}", presignedURL);
            return presignedURL;

        } catch (S3Exception e) {
            logger.error("Failed to generate presigned URL: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode(), e);
            throw e;
        }
    }
```

- Per i dettagli sull'API, consulta la sezione API GetObjectReference AWS SDK for Java 2.x .

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.

Ogni esempio include un collegamento al codice sorgente completo, dove puoi 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, CreateVaultconsulta 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, DeleteArchiveconsulta 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, DeleteVaultconsulta 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, ListVaultsconsulta 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, UploadArchiveconsulta AWS SDK for Java 2.x API Reference.

SageMaker Esempi di intelligenza artificiale 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 AI.

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.

Gli scenari sono esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio o combinate con altri Servizi AWS.

Ogni esempio include un collegamento al codice sorgente completo, in cui è possibile trovare istruzioni su come configurare ed eseguire il codice nel contesto.

Nozioni di base

Ciao SageMaker AI

I seguenti esempi di codice mostrano come iniziare a usare l' SageMaker IA.

SDK per Java 2.x

> **Note**
>
> C'è 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
 */
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, ListNotebookInstancesconsulta 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, CreatePipelineconsulta 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, DeletePipelineconsulta 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, DescribePipelineExecutionconsulta 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, StartPipelineExecutionconsulta 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.
- Eliminare le risorse.

Per ulteriori informazioni, consulta Creare ed eseguire SageMaker pipeline utilizzando AWS SDKs 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 =
sqsClient.getQueueAttributes(attributesRequest);
        Map<String, String> queueAtts = response.attributesAsStrings();
        for (Map.Entry<String, String> queueAtt : queueAtts.entrySet()) {
            System.out.println("Key = " + queueAtt.getKey() + ", Value = " +
queueAtt.getValue());
            queueArn = queueAtt.getValue();
        }

        CreateEventSourceMappingRequest eventSourceMappingRequest =
CreateEventSourceMappingRequest.builder()
                .eventSourceArn(queueArn)
                .functionName(lambdaName)
                .build();

        CreateEventSourceMappingResponse response1 =
lambdaClient.createEventSourceMapping(eventSourceMappingRequest);
        eventSourceMapping = response1.uuid();
```

```
        System.out.println("The mapping between the event source and Lambda function
was successful");
    }

    // Create an AWS Lambda function.
    public static String createLambdaFunction(LambdaClient awsLambda, String
functionName, String filePath, String role,
            String handler) {
        try {
            LambdaWaiter waiter = awsLambda.waiter();
            InputStream is = new FileInputStream(filePath);
            SdkBytes fileToUpload = SdkBytes.fromInputStream(is);
            FunctionCode code = FunctionCode.builder()
                    .zipFile(fileToUpload)
                    .build();

            CreateFunctionRequest functionRequest = CreateFunctionRequest.builder()
                    .functionName(functionName)
                    .description("SageMaker example function.")
                    .code(code)
                    .handler(handler)
                    .runtime(Runtime.JAVA11)
                    .timeout(200)
                    .memorySize(1024)
                    .role(role)
                    .build();

            // Create a Lambda function using a waiter.
            CreateFunctionResponse functionResponse =
awsLambda.createFunction(functionRequest);
            GetFunctionRequest getFunctionRequest = GetFunctionRequest.builder()
                    .functionName(functionName)
                    .build();
            WaiterResponse<GetFunctionResponse> waiterResponse =
waiter.waitUntilFunctionExists(getFunctionRequest);
            waiterResponse.matched().response().ifPresent(System.out::println);
            System.out.println("The function ARN is " +
functionResponse.functionArn());
            return functionResponse.functionArn();

        } catch (LambdaException | FileNotFoundException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
```

```
        return "";
    }

    public static String createSageMakerRole(IamClient iam, String roleName) {
        String[] sageMakerRolePolicies = getSageMakerRolePolicies();
        System.out.println("Creating a role to use with SageMaker.");
        String assumeRolePolicy = "{" +
                "\"Version\": \"2012-10-17\"," +
                "\"Statement\": [{" +
                "\"Effect\": \"Allow\"," +
                "\"Principal\": {" +
                "\"Service\": [" +
                "\"sagemaker.amazonaws.com\"," +
                "\"sagemaker-geospatial.amazonaws.com\"," +
                "\"lambda.amazonaws.com\"," +
                "\"s3.amazonaws.com\"" +
                "]" +
                "}," +
                "\"Action\": \"sts:AssumeRole\"" +
                "}]" +
                "}";

        try {
            CreateRoleRequest request = CreateRoleRequest.builder()
                    .roleName(roleName)
                    .assumeRolePolicyDocument(assumeRolePolicy)
                    .description("Created using the AWS SDK for Java")
                    .build();

            CreateRoleResponse roleResult = iam.createRole(request);

            // Attach the policies to the role.
            for (String policy : sageMakerRolePolicies) {
                AttachRolePolicyRequest attachRequest =
AttachRolePolicyRequest.builder()
                        .roleName(roleName)
                        .policyArn(policy)
                        .build();

                iam.attachRolePolicy(attachRequest);
            }

            // Allow time for the role to be ready.
            TimeUnit.SECONDS.sleep(15);
```

```
            System.out.println("Role ready with ARN " + roleResult.role().arn());
            return roleResult.role().arn();

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        } catch (InterruptedException e) {
            throw new RuntimeException(e);
        }
        return "";
    }

    private static String createLambdaRole(IamClient iam, String roleName) {
        String[] lambdaRolePolicies = getLambdaRolePolicies();
        String assumeRolePolicy = "{" +
                "\"Version\": \"2012-10-17\"," +
                "\"Statement\": [{" +
                "\"Effect\": \"Allow\"," +
                "\"Principal\": {" +
                "\"Service\": [" +
                "\"sagemaker.amazonaws.com\"," +
                "\"sagemaker-geospatial.amazonaws.com\"," +
                "\"lambda.amazonaws.com\"," +
                "\"s3.amazonaws.com\"" +
                "]" +
                "}," +
                "\"Action\": \"sts:AssumeRole\"" +
                "}]" +
                "}";

        try {
            CreateRoleRequest request = CreateRoleRequest.builder()
                    .roleName(roleName)
                    .assumeRolePolicyDocument(assumeRolePolicy)
                    .description("Created using the AWS SDK for Java")
                    .build();

            CreateRoleResponse roleResult = iam.createRole(request);

            // Attach the policies to the role.
            for (String policy : lambdaRolePolicies) {
                AttachRolePolicyRequest attachRequest =
AttachRolePolicyRequest.builder()
                        .roleName(roleName)
```

```
                        .policyArn(policy)
                        .build();

                iam.attachRolePolicy(attachRequest);
            }

            // Allow time for the role to be ready.
            TimeUnit.SECONDS.sleep(15);
            System.out.println("Role ready with ARN " + roleResult.role().arn());
            return roleResult.role().arn();

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());

        } catch (InterruptedException e) {
            throw new RuntimeException(e);
        }
        return "";
    }

    public static String checkFunction(LambdaClient lambdaClient, String
functionName, String filePath, String role,
            String handler) {
        System.out.println("Create an AWS Lambda function used in this workflow.");
        String functionArn;
        try {
            // Does this function already exist.
            GetFunctionRequest functionRequest = GetFunctionRequest.builder()
                    .functionName(functionName)
                    .build();

            GetFunctionResponse response =
lambdaClient.getFunction(functionRequest);
            functionArn = response.configuration().functionArn();

        } catch (LambdaException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            functionArn = createLambdaFunction(lambdaClient, functionName, filePath,
role, handler);
        }
        return functionArn;
    }

    // Check to see if the specific S3 bucket exists. If the S3 bucket exists, this
```

```
    // method returns true.
    public static boolean checkBucket(S3Client s3, String bucketName) {
        try {
            HeadBucketRequest headBucketRequest = HeadBucketRequest.builder()
                    .bucket(bucketName)
                    .build();

            s3.headBucket(headBucketRequest);
            System.out.println(bucketName + " exists");
            return true;

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }
        return false;
    }

    // Checks to see if the Amazon SQS queue exists. If not, this method creates a
    // new queue
    // and returns the ARN value.
    public static String checkQueue(SqsClient sqsClient, LambdaClient lambdaClient,
String queueName,
            String lambdaName) {
        System.out.println("Creating a queue for this use case.");
        String queueUrl;
        try {
            GetQueueUrlRequest request = GetQueueUrlRequest.builder()
                    .queueName(queueName)
                    .build();

            GetQueueUrlResponse response = sqsClient.getQueueUrl(request);
            queueUrl = response.queueUrl();
            System.out.println(queueUrl);

        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            queueUrl = setupQueue(sqsClient, lambdaClient, queueName, lambdaName);
        }
        return queueUrl;
    }

    // Checks to see if the Lambda role exists. If not, this method creates it.
    public static String checkLambdaRole(IamClient iam, String roleName) {
        System.out.println("Creating a role to for AWS Lambda to use.");
```

```
        String roleArn;
        try {
            GetRoleRequest roleRequest = GetRoleRequest.builder()
                    .roleName(roleName)
                    .build();

            GetRoleResponse response = iam.getRole(roleRequest);
            roleArn = response.role().arn();
            System.out.println(roleArn);

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            roleArn = createLambdaRole(iam, roleName);
        }
        return roleArn;
    }

    // Checks to see if the SageMaker role exists. If not, this method creates it.
    public static String checkSageMakerRole(IamClient iam, String roleName) {
        System.out.println("Creating a role 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.

Ogni esempio include un collegamento al codice sorgente completo, in cui 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, GetSecretValueconsulta 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.

Gli scenari sono esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio o combinate con altri Servizi AWS.

Ogni esempio include un collegamento al codice sorgente completo, dove puoi trovare istruzioni su come configurare ed eseguire il codice nel contesto.

Argomenti

- Azioni
- Scenari

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();

        listSESIdentities(client);
    }

    public static void listSESIdentities(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, ListIdentitiesconsulta 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, ListTemplatesconsulta 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.getStackTrace();
        }
    }

    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.getStackTrace();
        }
    }

    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, SendEmailconsulta 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, SendTemplatedEmailconsulta AWS SDK for Java 2.x API Reference.

Scenari

Creazione di un'applicazione Web per tracciare i dati DynamoDB

Il seguente esempio di codice mostra come creare un'applicazione Web che tiene traccia degli elementi di lavoro in una tabella Amazon DynamoDB e utilizza Amazon Simple Email Service (Amazon SES) per inviare report.

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, consulta l'esempio completo su. GitHub
>
> Servizi utilizzati in questo esempio
>
> - DynamoDB

- Amazon SES

Creazione di un'applicazione Web per tracciare i dati Amazon Redshift

Il seguente esempio di codice mostra come creare un'applicazione Web che traccia e genera report sugli elementi di lavoro utilizzando un database 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

Il seguente esempio di codice mostra come creare un'applicazione Web che tiene traccia degli elementi di lavoro in un database Amazon Aurora Serverless e utilizza Amazon Simple Email Service (Amazon SES) per inviare report.

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

Rilevamento dei DPI nelle immagini

Il seguente esempio di codice mostra come creare un'app che utilizza Amazon Rekognition per rilevare i dispositivi di protezione individuale (DPI) nelle immagini.

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

Rilevamento di oggetti nelle immagini

Il seguente esempio di codice mostra come creare un'app che utilizza Amazon Rekognition per rilevare oggetti per categoria nelle immagini.

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

Rilevamento di persone e oggetti in un video

Il seguente esempio di codice mostra come rilevare persone e oggetti in un video con Amazon Rekognition.

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
- Amazon SNS
- Amazon SQS

Utilizzo di Step Functions per richiamare le funzioni Lambda

Il seguente esempio di codice mostra come creare una macchina a AWS Step Functions stati che richiama AWS Lambda funzioni in sequenza.

SDK per Java 2.x

Mostra come creare un flusso di lavoro AWS senza server 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

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.

Gli scenari sono esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio o combinate con altri Servizi AWS.

Ogni esempio include un collegamento al codice sorgente completo, 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, CreateContactconsulta 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, CreateContactListconsulta 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, CreateEmailIdentityconsulta 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, CreateEmailTemplateconsulta 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, DeleteContactListconsulta 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, DeleteEmailIdentityconsulta 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, DeleteEmailTemplateconsulta 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, ListContactsconsulta 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.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * 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());
     }
```

Invia un messaggio con informazioni sull'intestazione.

```
public class SendwithHeader {

    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();

        String bodyHTML = """
                <html>
```

```
                    <head></head>
                    <body>
                        <h1>Hello!</h1>
                        <p>See the list of customers.</p>
                    </body>
                </html>
                """;

        sendWithHeader(sesv2Client, sender, recipient, subject, bodyHTML);
        sesv2Client.close();
    }

    /**
     * Sends an email using the AWS SES V2 client.
     *
     * @param sesv2Client the SES V2 client to use for sending the email
     * @param sender the email address of the sender
     * @param recipient the email address of the recipient
     * @param subject the subject of the email
     * @param bodyHTML the HTML content of the email body
     */
    public static void sendWithHeader(SesV2Client sesv2Client,
                                      String sender,
                                      String recipient,
                                      String subject,
                                      String bodyHTML) {
        EmailContent emailContent = EmailContent.builder()
                .simple(Message.builder()
                        .body(b -> b.html(c ->
 c.charset(UTF_8.name()).data(bodyHTML))
                                .text(c -> c.charset(UTF_8.name()).data(bodyHTML)))
                        .subject(c -> c.charset(UTF_8.name()).data(subject))
                        .headers(List.of(
                                MessageHeader.builder()
                                        .name("List-Unsubscribe")
                                        .value("<https://nutrition.co/?
address=x&topic=x>, <mailto:unsubscribe@nutrition.co?subject=TopicUnsubscribe>")
                                        .build(),
                                MessageHeader.builder()
                                        .name("List-Unsubscribe-Post")
                                        .value("List-Unsubscribe=One-Click")
                                        .build()))
                        .build())
                .build();
```

```
        SendEmailRequest request = SendEmailRequest.builder()
                .fromEmailAddress(sender)
                .destination(d -> d.toAddresses(recipient))
                .content(emailContent)
                .build();

        try {
            SendEmailResponse response = sesv2Client.sendEmail(request);
            System.out.println("Email sent! Message ID: " + response.messageId());
        } catch (SesV2Exception e) {
            System.err.println("Failed to send email: " +
 e.awsErrorDetails().errorMessage());
            throw new RuntimeException(e);
        }
    }
}
```

- Per i dettagli sull'API, consulta la sezione SendEmail AWS SDK for Java 2.xAPI Reference.

Scenari

Scenario della newsletter

Il seguente esempio di codice mostra come eseguire lo scenario di newsletter 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.

Gli scenari sono esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio o combinate con altri Servizi AWS.

Ogni esempio include un collegamento al codice sorgente completo, 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, ListTopicsconsulta 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, CheckIfPhoneNumberIsOptedOutconsulta 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 ConfirmSubscriptionsezione 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 CreateTopicsezione 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 DeleteTopicsezione 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 GetSMSAtrributes {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <topicArn>

                Where:
                   topicArn - The ARN of the topic from which to retrieve
 attributes.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String topicArn = args[0];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        getSNSAttrutes(snsClient, topicArn);
        snsClient.close();
    }

    public static void getSNSAttrutes(SnsClient snsClient, String topicArn) {
        try {
            GetSubscriptionAttributesRequest request =
 GetSubscriptionAttributesRequest.builder()
                    .subscriptionArn(topicArn)
                    .build();

            // Get the Subscription attributes
```

```
            GetSubscriptionAttributesResponse res =
 snsClient.getSubscriptionAttributes(request);
            Map<String, String> map = res.attributes();

            // Iterate through the map
            Iterator iter = map.entrySet().iterator();
            while (iter.hasNext()) {
                Map.Entry entry = (Map.Entry) iter.next();
                System.out.println("[Key] : " + entry.getKey() + " [Value] : " +
 entry.getValue());
            }

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }

        System.out.println("\n\nStatus was good");
    }
}
```

- Per i dettagli sull'API, consulta Get SMSAttributes in AWS SDK for Java 2.x API Reference.

`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 GetTopicAttributessezione 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 ListPhoneNumbersOptedOutsezione 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 ListSubscriptionssezione 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
\nTopics\n\n" + result.topics());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Per i dettagli sull'API, consulta la ListTopicssezione 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 i dettagli sull'API, consulta Set SMSAttributes in AWS SDK for Java 2.x API Reference.

`SetSubscriptionAttributes`

Il seguente esempio di codice mostra come utilizzare`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 SetSubscriptionAttributessezione 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 SetTopicAttributessezione 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 TagResourcesezione 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

Costruisci un'app per inviare dati a una tabella DynamoDB

Il seguente esempio di codice mostra come creare un'applicazione che invia dati a una tabella Amazon DynamoDB e ti avvisa quando un utente aggiorna la tabella.

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, consulta l'esempio completo su. GitHub

Servizi utilizzati in questo esempio

- DynamoDB
- Amazon SNS

Costruzione di un'applicazione Amazon SNS

Il seguente esempio di codice mostra come creare un'applicazione con funzionalità di sottoscrizione e pubblicazione e tradurre 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, consultate 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

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 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.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 device token or registration ID of the mobile device.
 This is a unique
               identifier provided by the device platform (e.g., Apple Push
 Notification Service (APNS) for iOS devices, Firebase Cloud Messaging (FCM)
               for Android devices) when the mobile app is registered to receive
 push notifications.

               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);
            return;
        }

        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());
        }
    }
}
```

Creazione di un'applicazione serverless per gestire foto

Nell'esempio di codice seguente viene illustrato come creare un'applicazione serverless che consente agli utenti di gestire le foto mediante 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 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",
                "FifoThroughputScope", "MessageGroup");

            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

Rilevamento di persone e oggetti in un video

Il seguente esempio di codice mostra come rilevare persone e oggetti in un video con Amazon Rekognition.

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
- Amazon SNS
- Amazon SQS

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 altro su 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);
        }
    }
}
```

Pubblicare 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 .

Pubblicazione di messaggi nelle code

L'esempio di codice seguente mostra come:

- Creazione di un argomento (FIFO o non FIFO).
- Sottoscrizione di diverse code all'argomento con la possibilità di applicare un filtro.
- Pubblicazione di un messaggio nell'argomento.
- Esame delle code per i messaggi ricevuti.

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.

```
package com.example.sns;

import
 software.amazon.awssdk.auth.credentials.EnvironmentVariableCredentialsProvider;
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.DeleteTopicRequest;
import software.amazon.awssdk.services.sns.model.DeleteTopicResponse;
import software.amazon.awssdk.services.sns.model.MessageAttributeValue;
import software.amazon.awssdk.services.sns.model.PublishRequest;
import software.amazon.awssdk.services.sns.model.PublishResponse;
import software.amazon.awssdk.services.sns.model.SetSubscriptionAttributesRequest;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.model.SubscribeRequest;
import software.amazon.awssdk.services.sns.model.SubscribeResponse;
import software.amazon.awssdk.services.sns.model.UnsubscribeRequest;
import software.amazon.awssdk.services.sns.model.UnsubscribeResponse;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.CreateQueueRequest;
import software.amazon.awssdk.services.sqs.model.DeleteMessageBatchRequest;
import software.amazon.awssdk.services.sqs.model.DeleteMessageBatchRequestEntry;
import software.amazon.awssdk.services.sqs.model.DeleteQueueRequest;
import software.amazon.awssdk.services.sqs.model.GetQueueAttributesRequest;
import software.amazon.awssdk.services.sqs.model.GetQueueAttributesResponse;
import software.amazon.awssdk.services.sqs.model.GetQueueUrlRequest;
import software.amazon.awssdk.services.sqs.model.GetQueueUrlResponse;
import software.amazon.awssdk.services.sqs.model.Message;
import software.amazon.awssdk.services.sqs.model.QueueAttributeName;
import software.amazon.awssdk.services.sqs.model.ReceiveMessageRequest;
import software.amazon.awssdk.services.sqs.model.SetQueueAttributesRequest;
import software.amazon.awssdk.services.sqs.model.SqsException;
```

```
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.Scanner;

import com.google.gson.Gson;
import com.google.gson.JsonArray;
import com.google.gson.JsonObject;
import com.google.gson.JsonPrimitive;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 * <p>
 * This Java example performs these tasks:
 * <p>
 * 1. Gives the user three options to choose from.
 * 2. Creates an Amazon Simple Notification Service (Amazon SNS) topic.
 * 3. Creates an Amazon Simple Queue Service (Amazon SQS) queue.
 * 4. Gets the SQS queue Amazon Resource Name (ARN) attribute.
 * 5. Attaches an AWS Identity and Access Management (IAM) policy to the queue.
 * 6. Subscribes to the SQS queue.
 * 7. Publishes a message to the topic.
 * 8. Displays the messages.
 * 9. Deletes the received message.
 * 10. Unsubscribes from the topic.
 * 11. Deletes the SNS topic.
 */
public class SNSWorkflow {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    public static void main(String[] args) {
        final String usage = "\n" +
            "Usage:\n" +
            "    <fifoQueueARN>\n\n" +
            "Where:\n" +
            "    accountId - Your AWS account Id value.";

        if (args.length != 1) {
```

```
            System.out.println(usage);
            System.exit(1);
        }

        SnsClient snsClient = SnsClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(EnvironmentVariableCredentialsProvider.create())
            .build();

        SqsClient sqsClient = SqsClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(EnvironmentVariableCredentialsProvider.create())
            .build();

        Scanner in = new Scanner(System.in);
        String accountId = args[0];
        String useFIFO;
        String duplication = "n";
        String topicName;
        String deduplicationID = null;
        String groupId = null;

        String topicArn;
        String sqsQueueName;
        String sqsQueueUrl;
        String sqsQueueArn;
        String subscriptionArn;
        boolean selectFIFO = false;

        String message;
        List<Message> messageList;
        List<String> filterList = new ArrayList<>();
        String msgAttValue = "";

        System.out.println(DASHES);
        System.out.println("Welcome to messaging with topics and queues.");
        System.out.println("In this scenario, you will create an SNS topic and
subscribe an SQS queue to the topic.\n" +
            "You can select from several options for configuring the topic and the
subscriptions for the queue.\n" +
            "You can then post to the topic and see the results in the queue.");
        System.out.println(DASHES);

        System.out.println(DASHES);
```

```
        System.out.println("SNS topics can be configured as FIFO (First-In-First-
Out).\n" +
            "FIFO topics deliver messages in order and support deduplication and
 message filtering.\n" +
            "Would you like to work with FIFO topics? (y/n)");
        useFIFO = in.nextLine();
        if (useFIFO.compareTo("y") == 0) {
            selectFIFO = true;
            System.out.println("You have selected FIFO");
            System.out.println(" Because you have chosen a FIFO topic, deduplication
 is supported.\n" +
                "        Deduplication IDs are either set in the message or
 automatically generated from content using a hash function.\n"
                +
                "        If a message is successfully published to an SNS FIFO
 topic, any message published and determined to have the same deduplication ID,\n"
                +
                "        within the five-minute deduplication interval, is accepted
 but not delivered.\n" +
                "        For more information about deduplication, see https://
docs.aws.amazon.com/sns/latest/dg/fifo-message-dedup.html.");

            System.out.println(
                "Would you like to use content-based deduplication instead of
 entering a deduplication ID? (y/n)");
            duplication = in.nextLine();
            if (duplication.compareTo("y") == 0) {
                System.out.println("Please enter a group id value");
                groupId = in.nextLine();
            } else {
                System.out.println("Please enter deduplication Id value");
                deduplicationID = in.nextLine();
                System.out.println("Please enter a group id value");
                groupId = in.nextLine();
            }
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("2. Create a topic.");
        System.out.println("Enter a name for your SNS topic.");
        topicName = in.nextLine();
        if (selectFIFO) {
```

```
            System.out.println("Because you have selected a FIFO topic, '.fifo' must
be appended to the topic name.");
            topicName = topicName + ".fifo";
            System.out.println("The name of the topic is " + topicName);
            topicArn = createFIFO(snsClient, topicName, duplication);
            System.out.println("The ARN of the FIFO topic is " + topicArn);

        } else {
            System.out.println("The name of the topic is " + topicName);
            topicArn = createSNSTopic(snsClient, topicName);
            System.out.println("The ARN of the non-FIFO topic is " + topicArn);

        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. Create an SQS queue.");
        System.out.println("Enter a name for your SQS queue.");
        sqsQueueName = in.nextLine();
        if (selectFIFO) {
            sqsQueueName = sqsQueueName + ".fifo";
        }
        sqsQueueUrl = createQueue(sqsClient, sqsQueueName, selectFIFO);
        System.out.println("The queue URL is " + sqsQueueUrl);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. Get the SQS queue ARN attribute.");
        sqsQueueArn = getSQSQueueAttrs(sqsClient, sqsQueueUrl);
        System.out.println("The ARN of the new queue is " + sqsQueueArn);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. Attach an IAM policy to the queue.");

        // Define the policy to use. Make sure that you change the REGION if you are
        // running this code
        // in a different region.
        String policy = """
        {
             "Statement": [
             {
                 "Effect": "Allow",
                         "Principal": {
```

```
                    "Service": "sns.amazonaws.com"
                },
                "Action": "sqs:SendMessage",
                        "Resource": "arn:aws:sqs:us-east-1:%s:%s",
                        "Condition": {
                    "ArnEquals": {
                        "aws:SourceArn": "arn:aws:sns:us-east-1:%s:%s"
                    }
                }
            }
            ]
        }
        """.formatted(accountId, sqsQueueName, accountId, topicName);

        setQueueAttr(sqsClient, sqsQueueUrl, policy);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6. Subscribe to the SQS queue.");
        if (selectFIFO) {
            System.out.println(
                "If you add a filter to this subscription, then only the filtered
 messages will be received in the queue.\n"
                    +
                    "For information about message filtering, see https://
docs.aws.amazon.com/sns/latest/dg/sns-message-filtering.html\n"
                    +
                    "For this example, you can filter messages by a \"tone\"
 attribute.");
            System.out.println("Would you like to filter messages for " +
 sqsQueueName + "'s subscription to the topic "
                + topicName + "?  (y/n)");
            String filterAns = in.nextLine();
            if (filterAns.compareTo("y") == 0) {
                boolean moreAns = false;
                System.out.println("You can filter messages by one or more of the
 following \"tone\" attributes.");
                System.out.println("1. cheerful");
                System.out.println("2. funny");
                System.out.println("3. serious");
                System.out.println("4. sincere");
                while (!moreAns) {
                    System.out.println("Select a number or choose 0 to end.");
                    String ans = in.nextLine();
```

```
                    switch (ans) {
                        case "1":
                            filterList.add("cheerful");
                            break;
                        case "2":
                            filterList.add("funny");
                            break;
                        case "3":
                            filterList.add("serious");
                            break;
                        case "4":
                            filterList.add("sincere");
                            break;
                        default:
                            moreAns = true;
                            break;
                    }
                }
            }
        }
        subscriptionArn = subQueue(snsClient, topicArn, sqsQueueArn, filterList);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Publish a message to the topic.");
        if (selectFIFO) {
            System.out.println("Would you like to add an attribute to this message?
(y/n)");
            String msgAns = in.nextLine();
            if (msgAns.compareTo("y") == 0) {
                System.out.println("You can filter messages by one or more of the
following \"tone\" attributes.");
                System.out.println("1. cheerful");
                System.out.println("2. funny");
                System.out.println("3. serious");
                System.out.println("4. sincere");
                System.out.println("Select a number or choose 0 to end.");
                String ans = in.nextLine();
                switch (ans) {
                    case "1":
                        msgAttValue = "cheerful";
                        break;
                    case "2":
                        msgAttValue = "funny";
```

```
                        break;
                    case "3":
                        msgAttValue = "serious";
                        break;
                    default:
                        msgAttValue = "sincere";
                        break;
                }

                System.out.println("Selected value is " + msgAttValue);
            }
            System.out.println("Enter a message.");
            message = in.nextLine();
            pubMessageFIFO(snsClient, message, topicArn, msgAttValue, duplication,
groupId, deduplicationID);

        } else {
            System.out.println("Enter a message.");
            message = in.nextLine();
            pubMessage(snsClient, message, topicArn);
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("8. Display the message. Press any key to continue.");
        in.nextLine();
        messageList = receiveMessages(sqsClient, sqsQueueUrl, msgAttValue);
        for (Message mes : messageList) {
            System.out.println("Message Id: " + mes.messageId());
            System.out.println("Full Message: " + mes.body());
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("9. Delete the received message. Press any key to
continue.");
        in.nextLine();
        deleteMessages(sqsClient, sqsQueueUrl, messageList);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("10. Unsubscribe from the topic and delete the queue.
Press any key to continue.");
        in.nextLine();
```

```
        unSub(snsClient, subscriptionArn);
        deleteSQSQueue(sqsClient, sqsQueueName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("11. Delete the topic. Press any key to continue.");
        in.nextLine();
        deleteSNSTopic(snsClient, topicArn);

        System.out.println(DASHES);
        System.out.println("The SNS/SQS workflow has completed successfully.");
        System.out.println(DASHES);
    }

    public static void deleteSNSTopic(SnsClient snsClient, String topicArn) {
        try {
            DeleteTopicRequest request = DeleteTopicRequest.builder()
                .topicArn(topicArn)
                .build();

            DeleteTopicResponse result = snsClient.deleteTopic(request);
            System.out.println("Status was " +
result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    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);
            System.out.println(queueName + " was successfully deleted.");
```

```
        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void unSub(SnsClient snsClient, String subscriptionArn) {
        try {
            UnsubscribeRequest request = UnsubscribeRequest.builder()
                .subscriptionArn(subscriptionArn)
                .build();

            UnsubscribeResponse result = snsClient.unsubscribe(request);
            System.out.println("Status was " + result.sdkHttpResponse().statusCode()
                + "\nSubscription was removed for " + request.subscriptionArn());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void deleteMessages(SqsClient sqsClient, String queueUrl,
List<Message> messages) {
        try {
            List<DeleteMessageBatchRequestEntry> entries = new ArrayList<>();
            for (Message msg : messages) {
                DeleteMessageBatchRequestEntry entry =
DeleteMessageBatchRequestEntry.builder()
                    .id(msg.messageId())
                    .build();

                entries.add(entry);
            }

            DeleteMessageBatchRequest deleteMessageBatchRequest =
DeleteMessageBatchRequest.builder()
                .queueUrl(queueUrl)
                .entries(entries)
                .build();

            sqsClient.deleteMessageBatch(deleteMessageBatchRequest);
            System.out.println("The batch delete of messages was successful");
```

```
        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static List<Message> receiveMessages(SqsClient sqsClient, String
queueUrl, String msgAttValue) {
        try {
            if (msgAttValue.isEmpty()) {
                ReceiveMessageRequest receiveMessageRequest =
ReceiveMessageRequest.builder()
                    .queueUrl(queueUrl)
                    .maxNumberOfMessages(5)
                    .build();
                return sqsClient.receiveMessage(receiveMessageRequest).messages();
            } else {
                // We know there are filters on the message.
                ReceiveMessageRequest receiveRequest =
ReceiveMessageRequest.builder()
                    .queueUrl(queueUrl)
                    .messageAttributeNames(msgAttValue) // Include other message
attributes if needed.
                    .maxNumberOfMessages(5)
                    .build();

                return sqsClient.receiveMessage(receiveRequest).messages();
            }

        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return null;
    }

    public static void pubMessage(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);
        }
    }

    public static void pubMessageFIFO(SnsClient snsClient,
                                      String message,
                                      String topicArn,
                                      String msgAttValue,
                                      String duplication,
                                      String groupId,
                                      String deduplicationID) {

        try {
            PublishRequest request;
            // Means the user did not choose to use a message attribute.
            if (msgAttValue.isEmpty()) {
                if (duplication.compareTo("y") == 0) {
                    request = PublishRequest.builder()
                        .message(message)
                        .messageGroupId(groupId)
                        .topicArn(topicArn)
                        .build();
                } else {
                    request = PublishRequest.builder()
                        .message(message)
                        .messageDeduplicationId(deduplicationID)
                        .messageGroupId(groupId)
                        .topicArn(topicArn)
                        .build();
                }

            } else {
                Map<String, MessageAttributeValue> messageAttributes = new
HashMap<>();
                messageAttributes.put(msgAttValue, MessageAttributeValue.builder()
                    .dataType("String")
                    .stringValue("true")
```

```
                    .build());

                if (duplication.compareTo("y") == 0) {
                    request = PublishRequest.builder()
                        .message(message)
                        .messageGroupId(groupId)
                        .topicArn(topicArn)
                        .build();
                } else {
                    // Create a publish request with the message and attributes.
                    request = PublishRequest.builder()
                        .topicArn(topicArn)
                        .message(message)
                        .messageDeduplicationId(deduplicationID)
                        .messageGroupId(groupId)
                        .messageAttributes(messageAttributes)
                        .build();
                }
            }

            // Publish the message to the topic.
            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);
        }
    }

    // Subscribe to the SQS queue.
    public static String subQueue(SnsClient snsClient, String topicArn, String
queueArn, List<String> filterList) {
        try {
            SubscribeRequest request;
            if (filterList.isEmpty()) {
                // No filter subscription is added.
                request = SubscribeRequest.builder()
                    .protocol("sqs")
                    .endpoint(queueArn)
                    .returnSubscriptionArn(true)
                    .topicArn(topicArn)
```

```
                    .build();

                SubscribeResponse result = snsClient.subscribe(request);
                System.out.println("The queue " + queueArn + " has been subscribed
to the topic " + topicArn + "\n" +
                    "with the subscription ARN " + result.subscriptionArn());
                return result.subscriptionArn();
            } else {
                request = SubscribeRequest.builder()
                    .protocol("sqs")
                    .endpoint(queueArn)
                    .returnSubscriptionArn(true)
                    .topicArn(topicArn)
                    .build();

                SubscribeResponse result = snsClient.subscribe(request);
                System.out.println("The queue " + queueArn + " has been subscribed
to the topic " + topicArn + "\n" +
                    "with the subscription ARN " + result.subscriptionArn());

                String attributeName = "FilterPolicy";
                Gson gson = new Gson();
                String jsonString = "{\"tone\": []}";
                JsonObject jsonObject = gson.fromJson(jsonString, JsonObject.class);
                JsonArray toneArray = jsonObject.getAsJsonArray("tone");
                for (String value : filterList) {
                    toneArray.add(new JsonPrimitive(value));
                }

                String updatedJsonString = gson.toJson(jsonObject);
                System.out.println(updatedJsonString);
                SetSubscriptionAttributesRequest attRequest =
SetSubscriptionAttributesRequest.builder()
                    .subscriptionArn(result.subscriptionArn())
                    .attributeName(attributeName)
                    .attributeValue(updatedJsonString)
                    .build();

                snsClient.setSubscriptionAttributes(attRequest);
                return result.subscriptionArn();
            }

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
```

```
            System.exit(1);
        }
        return "";
    }

    // Attach a policy to the queue.
    public static void setQueueAttr(SqsClient sqsClient, String queueUrl, String
policy) {
        try {
            Map<software.amazon.awssdk.services.sqs.model.QueueAttributeName,
String> attrMap = new HashMap<>();
            attrMap.put(QueueAttributeName.POLICY, policy);

            SetQueueAttributesRequest attributesRequest =
SetQueueAttributesRequest.builder()
                .queueUrl(queueUrl)
                .attributes(attrMap)
                .build();

            sqsClient.setQueueAttributes(attributesRequest);
            System.out.println("The policy has been successfully attached.");

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static String getSQSQueueAttrs(SqsClient sqsClient, String queueUrl) {
        // Specify the attributes to retrieve.
        List<QueueAttributeName> atts = new ArrayList<>();
        atts.add(QueueAttributeName.QUEUE_ARN);

        GetQueueAttributesRequest attributesRequest =
GetQueueAttributesRequest.builder()
            .queueUrl(queueUrl)
            .attributeNames(atts)
            .build();

        GetQueueAttributesResponse response =
sqsClient.getQueueAttributes(attributesRequest);
        Map<String, String> queueAtts = response.attributesAsStrings();
        for (Map.Entry<String, String> queueAtt : queueAtts.entrySet())
            return queueAtt.getValue();
```

```
        return "";
    }

    public static String createQueue(SqsClient sqsClient, String queueName, Boolean
selectFIFO) {
        try {
            System.out.println("\nCreate Queue");
            if (selectFIFO) {
                Map<QueueAttributeName, String> attrs = new HashMap<>();
                attrs.put(QueueAttributeName.FIFO_QUEUE, "true");
                CreateQueueRequest createQueueRequest = CreateQueueRequest.builder()
                    .queueName(queueName)
                    .attributes(attrs)
                    .build();

                sqsClient.createQueue(createQueueRequest);
                System.out.println("\nGet queue url");
                GetQueueUrlResponse getQueueUrlResponse = sqsClient

.getQueueUrl(GetQueueUrlRequest.builder().queueName(queueName).build());
                return getQueueUrlResponse.queueUrl();
            } else {
                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 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 "";
    }

    public static String createFIFO(SnsClient snsClient, String topicName, String
 duplication) {
        try {
            // Create a FIFO topic by using the SNS service client.
            Map<String, String> topicAttributes = new HashMap<>();
            if (duplication.compareTo("n") == 0) {
                topicAttributes.put("FifoTopic", "true");
                topicAttributes.put("ContentBasedDeduplication", "false");
            } else {
                topicAttributes.put("FifoTopic", "true");
                topicAttributes.put("ContentBasedDeduplication", "true");
            }

            CreateTopicRequest topicRequest = CreateTopicRequest.builder()
                .name(topicName)
                .attributes(topicAttributes)
                .build();

            CreateTopicResponse response = snsClient.createTopic(topicRequest);
            return response.topicArn();

        } catch (SnsException 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 .
  - CreateQueue
  - CreateTopic
  - DeleteMessageBatch
  - DeleteQueue
  - DeleteTopic
  - GetQueueAttributes
  - Pubblicare
  - ReceiveMessage
  - SetQueueAttributes
  - Subscribe
  - Unsubscribe

Utilizzo di un'API Gateway per richiamare una funzione Lambda

Il seguente esempio di codice mostra come creare una AWS Lambda funzione richiamata da Amazon API Gateway.

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 degli eventi pianificati per richiamare una funzione Lambda

Il seguente esempio di codice mostra come creare una AWS Lambda funzione richiamata da un evento EventBridge pianificato di Amazon.

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

- CloudWatch Registri
- DynamoDB
- EventBridge
- Lambda
- Amazon SNS

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.

Utilizzo di un evento SNS 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.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.

Gli scenari sono esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio o combinate con altri Servizi AWS.

Ogni esempio include un collegamento al codice sorgente completo, 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 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.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)
                    .maxNumberOfMessages(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 CreateQueuesezione 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, consulta la DeleteMessagesezione 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, consulta la DeleteQueuesezione 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, consulta la GetQueueUrlsezione 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, consulta la ListQueuessezione 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)
                    .maxNumberOfMessages(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, consulta la ReceiveMessagesezione 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.

Seguono due esempi di `SendMessage` operazione:

- Inviare un messaggio con un corpo e un ritardo

- Inviare un messaggio con un corpo e gli attributi del messaggio

Invia un messaggio con un corpo e un ritardo.

```
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);
        }
    }
}
```

Invia un messaggio con un corpo e gli attributi del messaggio.

```
    /**
     * <p>This method demonstrates how to add message attributes to a message.
     * Each attribute must specify a name, value, and data type. You use a Java Map
 to supply the attributes. The map's
     * key is the attribute name, and you specify the map's entry value using a
 builder that includes the attribute
     * value and data type.</p>
     *
     * <p>The data type must start with one of "String", "Number" or "Binary". You
 can optionally
     * define a custom extension by using a "." and your extension.</p>
```

```
     *
     * <p>The SQS Developer Guide provides more information on @see <a
     * href="https://docs.aws.amazon.com/AWSSimpleQueueService/latest/
SQSDeveloperGuide/sqs-message-metadata.html#sqs-message-attributes">message
     * attributes</a>.</p>
     *
     * @param thumbailPath Filesystem path of the image.
     * @param queueUrl     URL of the SQS queue.
     */
    static void sendMessageWithAttributes(Path thumbailPath, String queueUrl) {
        Map<String, MessageAttributeValue> messageAttributeMap;
        try {
            messageAttributeMap = Map.of(
                    "Name", MessageAttributeValue.builder()
                            .stringValue("Jane Doe")
                            .dataType("String").build(),
                    "Age", MessageAttributeValue.builder()
                            .stringValue("42")
                            .dataType("Number.int").build(),
                    "Image", MessageAttributeValue.builder()

 .binaryValue(SdkBytes.fromByteArray(Files.readAllBytes(thumbailPath)))
                            .dataType("Binary.jpg").build()
            );
        } catch (IOException e) {
            LOGGER.error("An I/O exception occurred reading thumbnail image: {}",
 e.getMessage(), e);
            throw new RuntimeException(e);
        }

        SendMessageRequest request = SendMessageRequest.builder()
                .queueUrl(queueUrl)
                .messageBody("Hello SQS")
                .messageAttributes(messageAttributeMap)
                .build();
        try {
            SendMessageResponse sendMessageResponse =
 SQS_CLIENT.sendMessage(request);
            LOGGER.info("Message ID: {}", sendMessageResponse.messageId());
        } catch (SqsException e) {
            LOGGER.error("Exception occurred sending message: {}", e.getMessage(),
 e);
            throw new RuntimeException(e);
        }
```

```
    }
```

- Per i dettagli sull'API, consulta la SendMessagesezione 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, consulta la SendMessageBatchsezione AWS SDK for Java 2.x API Reference.

`SetQueueAttributes`

Il seguente esempio di codice mostra come utilizzare`SetQueueAttributes`.

SDK per Java 2.x

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

Configura un Amazon SQS per utilizzare la crittografia lato server (SSE) utilizzando una chiave KMS personalizzata.

```
    public static void addEncryption(String queueName, String kmsMasterKeyAlias) {
        SqsClient sqsClient = SqsClient.create();

        GetQueueUrlRequest urlRequest = GetQueueUrlRequest.builder()
                .queueName(queueName)
                .build();

        GetQueueUrlResponse getQueueUrlResponse;
        try {
            getQueueUrlResponse = sqsClient.getQueueUrl(urlRequest);
        } catch (QueueDoesNotExistException e) {
            LOGGER.error(e.getMessage(), e);
            throw new RuntimeException(e);
        }
        String queueUrl = getQueueUrlResponse.queueUrl();


        Map<QueueAttributeName, String> attributes = Map.of(
                QueueAttributeName.KMS_MASTER_KEY_ID, kmsMasterKeyAlias,
                QueueAttributeName.KMS_DATA_KEY_REUSE_PERIOD_SECONDS, "140" // Set
the data key reuse period to 140 seconds.
        );                                                                  // This
is how long SQS can reuse the data key before requesting a new one from KMS.

        SetQueueAttributesRequest attRequest = SetQueueAttributesRequest.builder()
                .queueUrl(queueUrl)
                .attributes(attributes)
                .build();
        try {
            sqsClient.setQueueAttributes(attRequest);
            LOGGER.info("The attributes have been applied to {}", queueName);
```

```
        } catch (InvalidAttributeNameException | InvalidAttributeValueException e) {
            LOGGER.error(e.getMessage(), e);
            throw new RuntimeException(e);
        } finally {
            sqsClient.close();
        }
    }
```

- Per i dettagli sull'API, consulta la sezione API Reference. SetQueueAttributesAWS SDK for Java 2.x

Scenari

Crea un'applicazione di messaggistica

Il seguente esempio di codice mostra come creare un'applicazione di messaggistica 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 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",
                "FifoThroughputScope", "MessageGroup");

            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

Rilevamento di persone e oggetti in un video

Il seguente esempio di codice mostra come rilevare persone e oggetti in un video con Amazon Rekognition.

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
- Amazon SNS
- Amazon SQS

Gestisci messaggi di grandi dimensioni con S3

Il seguente esempio di codice mostra come utilizzare la Amazon SQS Extended Client Library per lavorare con messaggi Amazon SQS di grandi dimensioni.

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.amazon.sqs.javamessaging.AmazonSQSExtendedClient;
import com.amazon.sqs.javamessaging.ExtendedClientConfiguration;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.joda.time.DateTime;
import org.joda.time.format.DateTimeFormat;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.BucketLifecycleConfiguration;
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.ExpirationStatus;
import software.amazon.awssdk.services.s3.model.LifecycleExpiration;
import software.amazon.awssdk.services.s3.model.LifecycleRule;
import software.amazon.awssdk.services.s3.model.LifecycleRuleFilter;
import software.amazon.awssdk.services.s3.model.ListObjectVersionsRequest;
import software.amazon.awssdk.services.s3.model.ListObjectVersionsResponse;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Request;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Response;
import
 software.amazon.awssdk.services.s3.model.PutBucketLifecycleConfigurationRequest;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.CreateQueueRequest;
import software.amazon.awssdk.services.sqs.model.CreateQueueResponse;
import software.amazon.awssdk.services.sqs.model.DeleteMessageRequest;
import software.amazon.awssdk.services.sqs.model.DeleteQueueRequest;
import software.amazon.awssdk.services.sqs.model.Message;
import software.amazon.awssdk.services.sqs.model.ReceiveMessageRequest;
import software.amazon.awssdk.services.sqs.model.ReceiveMessageResponse;
import software.amazon.awssdk.services.sqs.model.SendMessageRequest;

import java.util.Arrays;
import java.util.List;
import java.util.UUID;

/**
 * Example of using Amazon SQS Extended Client Library for Java 2.x.
 */
public class SqsExtendedClientExample {
    private static final Logger logger =
 LoggerFactory.getLogger(SqsExtendedClientExample.class);

    private String s3BucketName;
    private String queueUrl;
    private final String queueName;
    private final S3Client s3Client;
    private final SqsClient sqsExtendedClient;
    private final int messageSize;

    /**
     * Constructor with default clients and message size.
     */
    public SqsExtendedClientExample() {
```

```
        this(S3Client.create(), 300000);
    }

    /**
     * Constructor with custom S3 client and message size.
     *
     * @param s3Client The S3 client to use
     * @param messageSize The size of the test message to create
     */
    public SqsExtendedClientExample(S3Client s3Client, int messageSize) {
        this.s3Client = s3Client;
        this.messageSize = messageSize;

        // Generate a unique bucket name.
        this.s3BucketName = UUID.randomUUID() + "-" +
                DateTimeFormat.forPattern("yyMMdd-hhmmss").print(new DateTime());

        // Generate a unique queue name.
        this.queueName = "MyQueue-" + UUID.randomUUID();

        // Configure the SQS extended client.
        final ExtendedClientConfiguration extendedClientConfig = new
ExtendedClientConfiguration()
                .withPayloadSupportEnabled(s3Client, s3BucketName);

        this.sqsExtendedClient = new
AmazonSQSExtendedClient(SqsClient.builder().build(), extendedClientConfig);
    }

    public static void main(String[] args) {
        SqsExtendedClientExample example = new SqsExtendedClientExample();
        try {
            example.setup();
            example.sendAndReceiveMessage();
        } finally {
            example.cleanup();
        }
    }

    /**
     * Send a large message and receive it back.
     *
     * @return The received message
     */
```

```
    public Message sendAndReceiveMessage() {
        try {
            // Create a large message.
            char[] chars = new char[messageSize];
            Arrays.fill(chars, 'x');
            String largeMessage = new String(chars);

            // Send the message.
            final SendMessageRequest sendMessageRequest =
SendMessageRequest.builder()
                    .queueUrl(queueUrl)
                    .messageBody(largeMessage)
                    .build();

            sqsExtendedClient.sendMessage(sendMessageRequest);
            logger.info("Sent message of size: {}", largeMessage.length());

            // Receive and return the message.
            final ReceiveMessageResponse receiveMessageResponse =
sqsExtendedClient.receiveMessage(
                    ReceiveMessageRequest.builder().queueUrl(queueUrl).build());

            List<Message> messages = receiveMessageResponse.messages();
            if (messages.isEmpty()) {
                throw new RuntimeException("No messages received");
            }

            Message message = messages.getFirst();
            logger.info("\nMessage received.");
            logger.info("  ID: {}", message.messageId());
            logger.info("  Receipt handle: {}", message.receiptHandle());
            logger.info("  Message body size: {}", message.body().length());
            logger.info("  Message body (first 5 characters): {}",
message.body().substring(0, 5));

            return message;
        } catch (RuntimeException e) {
            logger.error("Error during message processing: {}", e.getMessage(), e);
            throw e;
        }
    }
```

- 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 .
  - CreateBucket
  - PutBucketLifecycleConfiguration
  - ReceiveMessage
  - SendMessage

Elabora le notifiche degli eventi S3

Il seguente esempio di codice mostra come utilizzare le notifiche di eventi S3 in modo orientato agli oggetti.

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.

Questo esempio mostra come elaborare un evento di notifica S3 utilizzando Amazon SQS.

```
    /**
     * This method receives S3 event notifications by using an SqsAsyncClient.
     * After the client receives the messages it deserializes the JSON payload and
 logs them. It uses
     * the S3EventNotification class (part of the S3 event notification API for
 Java) to deserialize
     * the JSON payload and access the messages in an object-oriented way.
     *
     * @param queueUrl The URL of the AWS SQS queue that receives the S3 event
 notifications.
     * @see <a href="https://sdk.amazonaws.com/java/api/latest/software/amazon/
awssdk/eventnotifications/s3/model/package-summary.html">S3EventNotification API</
a>.
     * <p>
     * To use S3 event notification serialization/deserialization to objects, add
 the following
     * dependency to your Maven pom.xml file.
     * <dependency>
```

```
    * <groupId>software.amazon.awssdk</groupId>
    * <artifactId>s3-event-notifications</artifactId>
    * <version><LATEST></version>
    * </dependency>
    * <p>
    * The S3 event notification API became available with version 2.25.11 of the
Java SDK.
    * <p>
    * This example shows the use of the API with AWS SQS, but it can be used to
process S3 event notifications
    * in AWS SNS or AWS Lambda as well.
    * <p>
    * Note: The S3EventNotification class does not work with messages routed
through AWS EventBridge.
    */
   static void processS3Events(String bucketName, String queueUrl, String queueArn)
{
       try {
           // Configure the bucket to send Object Created and Object Tagging
notifications to an existing SQS queue.
           s3Client.putBucketNotificationConfiguration(b -> b
                   .notificationConfiguration(ncb -> ncb
                           .queueConfigurations(qcb -> qcb
                                   .events(Event.S3_OBJECT_CREATED,
Event.S3_OBJECT_TAGGING)
                                   .queueArn(queueArn)))
                           .bucket(bucketName)
           ).join();

           triggerS3EventNotifications(bucketName);
           // Wait for event notifications to propagate.
           Thread.sleep(Duration.ofSeconds(5).toMillis());

           boolean didReceiveMessages = true;
           while (didReceiveMessages) {
               // Display the number of messages that are available in the queue.
               sqsClient.getQueueAttributes(b -> b
                               .queueUrl(queueUrl)

.attributeNames(QueueAttributeName.APPROXIMATE_NUMBER_OF_MESSAGES)
                       ).thenAccept(attributeResponse ->
                               logger.info("Approximate number of messages in the
queue: {}",
```

```
attributeResponse.attributes().get(QueueAttributeName.APPROXIMATE_NUMBER_OF_MESSAGES)))
                        .join();

                // Receive the messages.
                ReceiveMessageResponse response = sqsClient.receiveMessage(b -> b
                        .queueUrl(queueUrl)
                ).get();
                logger.info("Count of received messages: {}",
response.messages().size());
                didReceiveMessages = !response.messages().isEmpty();

                // Create a collection to hold the received message for deletion
                // after we log the messages.
                HashSet<DeleteMessageBatchRequestEntry> messagesToDelete = new
HashSet<>();
                // Process each message.
                response.messages().forEach(message -> {
                    logger.info("Message id: {}", message.messageId());
                    // Deserialize JSON message body to a S3EventNotification object
                    // to access messages in an object-oriented way.
                    S3EventNotification event =
S3EventNotification.fromJson(message.body());

                    // Log the S3 event notification record details.
                    if (event.getRecords() != null) {
                        event.getRecords().forEach(record -> {
                            String eventName = record.getEventName();
                            String key = record.getS3().getObject().getKey();
                            logger.info(record.toString());
                            logger.info("Event name is {} and key is {}", eventName,
key);
                        });
                    }
                    // Add logged messages to collection for batch deletion.
                    messagesToDelete.add(DeleteMessageBatchRequestEntry.builder()
                            .id(message.messageId())
                            .receiptHandle(message.receiptHandle())
                            .build());
                });
                // Delete messages.
                if (!messagesToDelete.isEmpty()) {
                    sqsClient.deleteMessageBatch(DeleteMessageBatchRequest.builder()
                            .queueUrl(queueUrl)
```

```
                        .entries(messagesToDelete)
                        .build()
                ).join();
            }
        } // End of while block.
    } catch (InterruptedException | ExecutionException e) {
        throw new RuntimeException(e);
    }
}
```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella Documentazione di riferimento delle API AWS SDK for Java 2.x .
  - DeleteMessageBatch
  - GetQueueAttributes
  - PutBucketNotificationConfiguration
  - ReceiveMessage

Pubblicazione di messaggi nelle code

L'esempio di codice seguente mostra come:

- Creazione di un argomento (FIFO o non FIFO).
- Sottoscrizione di diverse code all'argomento con la possibilità di applicare un filtro.
- Pubblicazione di un messaggio nell'argomento.
- Esame delle code per i messaggi ricevuti.

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.

```
package com.example.sns;
```

```
import
 software.amazon.awssdk.auth.credentials.EnvironmentVariableCredentialsProvider;
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.DeleteTopicRequest;
import software.amazon.awssdk.services.sns.model.DeleteTopicResponse;
import software.amazon.awssdk.services.sns.model.MessageAttributeValue;
import software.amazon.awssdk.services.sns.model.PublishRequest;
import software.amazon.awssdk.services.sns.model.PublishResponse;
import software.amazon.awssdk.services.sns.model.SetSubscriptionAttributesRequest;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.model.SubscribeRequest;
import software.amazon.awssdk.services.sns.model.SubscribeResponse;
import software.amazon.awssdk.services.sns.model.UnsubscribeRequest;
import software.amazon.awssdk.services.sns.model.UnsubscribeResponse;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.CreateQueueRequest;
import software.amazon.awssdk.services.sqs.model.DeleteMessageBatchRequest;
import software.amazon.awssdk.services.sqs.model.DeleteMessageBatchRequestEntry;
import software.amazon.awssdk.services.sqs.model.DeleteQueueRequest;
import software.amazon.awssdk.services.sqs.model.GetQueueAttributesRequest;
import software.amazon.awssdk.services.sqs.model.GetQueueAttributesResponse;
import software.amazon.awssdk.services.sqs.model.GetQueueUrlRequest;
import software.amazon.awssdk.services.sqs.model.GetQueueUrlResponse;
import software.amazon.awssdk.services.sqs.model.Message;
import software.amazon.awssdk.services.sqs.model.QueueAttributeName;
import software.amazon.awssdk.services.sqs.model.ReceiveMessageRequest;
import software.amazon.awssdk.services.sqs.model.SetQueueAttributesRequest;
import software.amazon.awssdk.services.sqs.model.SqsException;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.Scanner;

import com.google.gson.Gson;
import com.google.gson.JsonArray;
import com.google.gson.JsonObject;
import com.google.gson.JsonPrimitive;

/**
```

```
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 * <p>
 * This Java example performs these tasks:
 * <p>
 * 1. Gives the user three options to choose from.
 * 2. Creates an Amazon Simple Notification Service (Amazon SNS) topic.
 * 3. Creates an Amazon Simple Queue Service (Amazon SQS) queue.
 * 4. Gets the SQS queue Amazon Resource Name (ARN) attribute.
 * 5. Attaches an AWS Identity and Access Management (IAM) policy to the queue.
 * 6. Subscribes to the SQS queue.
 * 7. Publishes a message to the topic.
 * 8. Displays the messages.
 * 9. Deletes the received message.
 * 10. Unsubscribes from the topic.
 * 11. Deletes the SNS topic.
 */
public class SNSWorkflow {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    public static void main(String[] args) {
        final String usage = "\n" +
            "Usage:\n" +
            "    <fifoQueueARN>\n\n" +
            "Where:\n" +
            "    accountId - Your AWS account Id value.";

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        SnsClient snsClient = SnsClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(EnvironmentVariableCredentialsProvider.create())
            .build();

        SqsClient sqsClient = SqsClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(EnvironmentVariableCredentialsProvider.create())
```

```
            .build();

        Scanner in = new Scanner(System.in);
        String accountId = args[0];
        String useFIFO;
        String duplication = "n";
        String topicName;
        String deduplicationID = null;
        String groupId = null;

        String topicArn;
        String sqsQueueName;
        String sqsQueueUrl;
        String sqsQueueArn;
        String subscriptionArn;
        boolean selectFIFO = false;

        String message;
        List<Message> messageList;
        List<String> filterList = new ArrayList<>();
        String msgAttValue = "";

        System.out.println(DASHES);
        System.out.println("Welcome to messaging with topics and queues.");
        System.out.println("In this scenario, you will create an SNS topic and
 subscribe an SQS queue to the topic.\n" +
            "You can select from several options for configuring the topic and the
 subscriptions for the queue.\n" +
            "You can then post to the topic and see the results in the queue.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("SNS topics can be configured as FIFO (First-In-First-
Out).\n" +
            "FIFO topics deliver messages in order and support deduplication and
 message filtering.\n" +
            "Would you like to work with FIFO topics? (y/n)");
        useFIFO = in.nextLine();
        if (useFIFO.compareTo("y") == 0) {
            selectFIFO = true;
            System.out.println("You have selected FIFO");
            System.out.println(" Because you have chosen a FIFO topic, deduplication
 is supported.\n" +
```

```
                "        Deduplication IDs are either set in the message or
 automatically generated from content using a hash function.\n"
                +
                "        If a message is successfully published to an SNS FIFO
 topic, any message published and determined to have the same deduplication ID,\n"
                +
                "        within the five-minute deduplication interval, is accepted
 but not delivered.\n" +
                "        For more information about deduplication, see https://
docs.aws.amazon.com/sns/latest/dg/fifo-message-dedup.html.");

            System.out.println(
                "Would you like to use content-based deduplication instead of
 entering a deduplication ID? (y/n)");
            duplication = in.nextLine();
            if (duplication.compareTo("y") == 0) {
                System.out.println("Please enter a group id value");
                groupId = in.nextLine();
            } else {
                System.out.println("Please enter deduplication Id value");
                deduplicationID = in.nextLine();
                System.out.println("Please enter a group id value");
                groupId = in.nextLine();
            }
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("2. Create a topic.");
        System.out.println("Enter a name for your SNS topic.");
        topicName = in.nextLine();
        if (selectFIFO) {
            System.out.println("Because you have selected a FIFO topic, '.fifo' must
 be appended to the topic name.");
            topicName = topicName + ".fifo";
            System.out.println("The name of the topic is " + topicName);
            topicArn = createFIFO(snsClient, topicName, duplication);
            System.out.println("The ARN of the FIFO topic is " + topicArn);

        } else {
            System.out.println("The name of the topic is " + topicName);
            topicArn = createSNSTopic(snsClient, topicName);
            System.out.println("The ARN of the non-FIFO topic is " + topicArn);
```

```
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. Create an SQS queue.");
        System.out.println("Enter a name for your SQS queue.");
        sqsQueueName = in.nextLine();
        if (selectFIFO) {
            sqsQueueName = sqsQueueName + ".fifo";
        }
        sqsQueueUrl = createQueue(sqsClient, sqsQueueName, selectFIFO);
        System.out.println("The queue URL is " + sqsQueueUrl);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. Get the SQS queue ARN attribute.");
        sqsQueueArn = getSQSQueueAttrs(sqsClient, sqsQueueUrl);
        System.out.println("The ARN of the new queue is " + sqsQueueArn);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. Attach an IAM policy to the queue.");

        // Define the policy to use. Make sure that you change the REGION if you are
        // running this code
        // in a different region.
        String policy = """
        {
             "Statement": [
             {
                 "Effect": "Allow",
                         "Principal": {
                     "Service": "sns.amazonaws.com"
                 },
                 "Action": "sqs:SendMessage",
                         "Resource": "arn:aws:sqs:us-east-1:%s:%s",
                         "Condition": {
                     "ArnEquals": {
                         "aws:SourceArn": "arn:aws:sns:us-east-1:%s:%s"
                     }
                 }
             }
             ]
         }
```

```
        """.formatted(accountId, sqsQueueName, accountId, topicName);

        setQueueAttr(sqsClient, sqsQueueUrl, policy);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6. Subscribe to the SQS queue.");
        if (selectFIFO) {
            System.out.println(
                "If you add a filter to this subscription, then only the filtered
 messages will be received in the queue.\n"
                    +
                    "For information about message filtering, see https://
docs.aws.amazon.com/sns/latest/dg/sns-message-filtering.html\n"
                    +
                    "For this example, you can filter messages by a \"tone\"
 attribute.");
            System.out.println("Would you like to filter messages for " +
 sqsQueueName + "'s subscription to the topic "
                + topicName + "?  (y/n)");
            String filterAns = in.nextLine();
            if (filterAns.compareTo("y") == 0) {
                boolean moreAns = false;
                System.out.println("You can filter messages by one or more of the
 following \"tone\" attributes.");
                System.out.println("1. cheerful");
                System.out.println("2. funny");
                System.out.println("3. serious");
                System.out.println("4. sincere");
                while (!moreAns) {
                    System.out.println("Select a number or choose 0 to end.");
                    String ans = in.nextLine();
                    switch (ans) {
                        case "1":
                            filterList.add("cheerful");
                            break;
                        case "2":
                            filterList.add("funny");
                            break;
                        case "3":
                            filterList.add("serious");
                            break;
                        case "4":
                            filterList.add("sincere");
```

```
                            break;
                        default:
                            moreAns = true;
                            break;
                    }
                }
            }
        }
        subscriptionArn = subQueue(snsClient, topicArn, sqsQueueArn, filterList);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Publish a message to the topic.");
        if (selectFIFO) {
            System.out.println("Would you like to add an attribute to this message?
(y/n)");
            String msgAns = in.nextLine();
            if (msgAns.compareTo("y") == 0) {
                System.out.println("You can filter messages by one or more of the
following \"tone\" attributes.");
                System.out.println("1. cheerful");
                System.out.println("2. funny");
                System.out.println("3. serious");
                System.out.println("4. sincere");
                System.out.println("Select a number or choose 0 to end.");
                String ans = in.nextLine();
                switch (ans) {
                    case "1":
                        msgAttValue = "cheerful";
                        break;
                    case "2":
                        msgAttValue = "funny";
                        break;
                    case "3":
                        msgAttValue = "serious";
                        break;
                    default:
                        msgAttValue = "sincere";
                        break;
                }

                System.out.println("Selected value is " + msgAttValue);
            }
            System.out.println("Enter a message.");
```

```
            message = in.nextLine();
            pubMessageFIFO(snsClient, message, topicArn, msgAttValue, duplication,
groupId, deduplicationID);

        } else {
            System.out.println("Enter a message.");
            message = in.nextLine();
            pubMessage(snsClient, message, topicArn);
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("8. Display the message. Press any key to continue.");
        in.nextLine();
        messageList = receiveMessages(sqsClient, sqsQueueUrl, msgAttValue);
        for (Message mes : messageList) {
            System.out.println("Message Id: " + mes.messageId());
            System.out.println("Full Message: " + mes.body());
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("9. Delete the received message. Press any key to
continue.");
        in.nextLine();
        deleteMessages(sqsClient, sqsQueueUrl, messageList);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("10. Unsubscribe from the topic and delete the queue.
Press any key to continue.");
        in.nextLine();
        unSub(snsClient, subscriptionArn);
        deleteSQSQueue(sqsClient, sqsQueueName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("11. Delete the topic. Press any key to continue.");
        in.nextLine();
        deleteSNSTopic(snsClient, topicArn);

        System.out.println(DASHES);
        System.out.println("The SNS/SQS workflow has completed successfully.");
        System.out.println(DASHES);
```

```
    }

    public static void deleteSNSTopic(SnsClient snsClient, String topicArn) {
        try {
            DeleteTopicRequest request = DeleteTopicRequest.builder()
                .topicArn(topicArn)
                .build();

            DeleteTopicResponse result = snsClient.deleteTopic(request);
            System.out.println("Status was " +
result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    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);
            System.out.println(queueName + " was successfully deleted.");

        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void unSub(SnsClient snsClient, String subscriptionArn) {
        try {
            UnsubscribeRequest request = UnsubscribeRequest.builder()
                .subscriptionArn(subscriptionArn)
                .build();
```

```
            UnsubscribeResponse result = snsClient.unsubscribe(request);
            System.out.println("Status was " + result.sdkHttpResponse().statusCode()
                + "\nSubscription was removed for " + request.subscriptionArn());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void deleteMessages(SqsClient sqsClient, String queueUrl,
List<Message> messages) {
        try {
            List<DeleteMessageBatchRequestEntry> entries = new ArrayList<>();
            for (Message msg : messages) {
                DeleteMessageBatchRequestEntry entry =
DeleteMessageBatchRequestEntry.builder()
                    .id(msg.messageId())
                    .build();

                entries.add(entry);
            }

            DeleteMessageBatchRequest deleteMessageBatchRequest =
DeleteMessageBatchRequest.builder()
                .queueUrl(queueUrl)
                .entries(entries)
                .build();

            sqsClient.deleteMessageBatch(deleteMessageBatchRequest);
            System.out.println("The batch delete of messages was successful");

        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static List<Message> receiveMessages(SqsClient sqsClient, String
queueUrl, String msgAttValue) {
        try {
            if (msgAttValue.isEmpty()) {
                ReceiveMessageRequest receiveMessageRequest =
ReceiveMessageRequest.builder()
```

```
                    .queueUrl(queueUrl)
                    .maxNumberOfMessages(5)
                    .build();
                return sqsClient.receiveMessage(receiveMessageRequest).messages();
            } else {
                // We know there are filters on the message.
                ReceiveMessageRequest receiveRequest =
ReceiveMessageRequest.builder()
                    .queueUrl(queueUrl)
                    .messageAttributeNames(msgAttValue) // Include other message
attributes if needed.
                    .maxNumberOfMessages(5)
                    .build();

                return sqsClient.receiveMessage(receiveRequest).messages();
            }

        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return null;
    }

    public static void pubMessage(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);
        }
    }

    public static void pubMessageFIFO(SnsClient snsClient,
```

```
                                    String message,
                                    String topicArn,
                                    String msgAttValue,
                                    String duplication,
                                    String groupId,
                                    String deduplicationID) {

        try {
            PublishRequest request;
            // Means the user did not choose to use a message attribute.
            if (msgAttValue.isEmpty()) {
                if (duplication.compareTo("y") == 0) {
                    request = PublishRequest.builder()
                        .message(message)
                        .messageGroupId(groupId)
                        .topicArn(topicArn)
                        .build();
                } else {
                    request = PublishRequest.builder()
                        .message(message)
                        .messageDeduplicationId(deduplicationID)
                        .messageGroupId(groupId)
                        .topicArn(topicArn)
                        .build();
                }

            } else {
                Map<String, MessageAttributeValue> messageAttributes = new
HashMap<>();
                messageAttributes.put(msgAttValue, MessageAttributeValue.builder()
                    .dataType("String")
                    .stringValue("true")
                    .build());

                if (duplication.compareTo("y") == 0) {
                    request = PublishRequest.builder()
                        .message(message)
                        .messageGroupId(groupId)
                        .topicArn(topicArn)
                        .build();
                } else {
                    // Create a publish request with the message and attributes.
                    request = PublishRequest.builder()
                        .topicArn(topicArn)
```

```
                        .message(message)
                        .messageDeduplicationId(deduplicationID)
                        .messageGroupId(groupId)
                        .messageAttributes(messageAttributes)
                        .build();
                }
            }

            // Publish the message to the topic.
            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);
        }
    }

    // Subscribe to the SQS queue.
    public static String subQueue(SnsClient snsClient, String topicArn, String
queueArn, List<String> filterList) {
        try {
            SubscribeRequest request;
            if (filterList.isEmpty()) {
                // No filter subscription is added.
                request = SubscribeRequest.builder()
                    .protocol("sqs")
                    .endpoint(queueArn)
                    .returnSubscriptionArn(true)
                    .topicArn(topicArn)
                    .build();

                SubscribeResponse result = snsClient.subscribe(request);
                System.out.println("The queue " + queueArn + " has been subscribed
to the topic " + topicArn + "\n" +
                    "with the subscription ARN " + result.subscriptionArn());
                return result.subscriptionArn();
            } else {
                request = SubscribeRequest.builder()
                    .protocol("sqs")
                    .endpoint(queueArn)
                    .returnSubscriptionArn(true)
```

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

                SubscribeResponse result = snsClient.subscribe(request);
                System.out.println("The queue " + queueArn + " has been subscribed
to the topic " + topicArn + "\n" +
                    "with the subscription ARN " + result.subscriptionArn());

                String attributeName = "FilterPolicy";
                Gson gson = new Gson();
                String jsonString = "{\"tone\": []}";
                JsonObject jsonObject = gson.fromJson(jsonString, JsonObject.class);
                JsonArray toneArray = jsonObject.getAsJsonArray("tone");
                for (String value : filterList) {
                    toneArray.add(new JsonPrimitive(value));
                }

                String updatedJsonString = gson.toJson(jsonObject);
                System.out.println(updatedJsonString);
                SetSubscriptionAttributesRequest attRequest =
SetSubscriptionAttributesRequest.builder()
                    .subscriptionArn(result.subscriptionArn())
                    .attributeName(attributeName)
                    .attributeValue(updatedJsonString)
                    .build();

                snsClient.setSubscriptionAttributes(attRequest);
                return result.subscriptionArn();
            }

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }

    // Attach a policy to the queue.
    public static void setQueueAttr(SqsClient sqsClient, String queueUrl, String
policy) {
        try {
            Map<software.amazon.awssdk.services.sqs.model.QueueAttributeName,
String> attrMap = new HashMap<>();
            attrMap.put(QueueAttributeName.POLICY, policy);
```

```
            SetQueueAttributesRequest attributesRequest =
SetQueueAttributesRequest.builder()
                .queueUrl(queueUrl)
                .attributes(attrMap)
                .build();

            sqsClient.setQueueAttributes(attributesRequest);
            System.out.println("The policy has been successfully attached.");

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static String getSQSQueueAttrs(SqsClient sqsClient, String queueUrl) {
        // Specify the attributes to retrieve.
        List<QueueAttributeName> atts = new ArrayList<>();
        atts.add(QueueAttributeName.QUEUE_ARN);

        GetQueueAttributesRequest attributesRequest =
GetQueueAttributesRequest.builder()
            .queueUrl(queueUrl)
            .attributeNames(atts)
            .build();

        GetQueueAttributesResponse response =
sqsClient.getQueueAttributes(attributesRequest);
        Map<String, String> queueAtts = response.attributesAsStrings();
        for (Map.Entry<String, String> queueAtt : queueAtts.entrySet())
            return queueAtt.getValue();

        return "";
    }

    public static String createQueue(SqsClient sqsClient, String queueName, Boolean
selectFIFO) {
        try {
            System.out.println("\nCreate Queue");
            if (selectFIFO) {
                Map<QueueAttributeName, String> attrs = new HashMap<>();
                attrs.put(QueueAttributeName.FIFO_QUEUE, "true");
                CreateQueueRequest createQueueRequest = CreateQueueRequest.builder()
```

```
                    .queueName(queueName)
                    .attributes(attrs)
                    .build();

                sqsClient.createQueue(createQueueRequest);
                System.out.println("\nGet queue url");
                GetQueueUrlResponse getQueueUrlResponse = sqsClient

.getQueueUrl(GetQueueUrlRequest.builder().queueName(queueName).build());
                return getQueueUrlResponse.queueUrl();
            } else {
                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 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 "";
    }

    public static String createFIFO(SnsClient snsClient, String topicName, String
 duplication) {
        try {
            // Create a FIFO topic by using the SNS service client.
            Map<String, String> topicAttributes = new HashMap<>();
            if (duplication.compareTo("n") == 0) {
                topicAttributes.put("FifoTopic", "true");
                topicAttributes.put("ContentBasedDeduplication", "false");
            } else {
                topicAttributes.put("FifoTopic", "true");
                topicAttributes.put("ContentBasedDeduplication", "true");
            }

            CreateTopicRequest topicRequest = CreateTopicRequest.builder()
                .name(topicName)
                .attributes(topicAttributes)
                .build();

            CreateTopicResponse response = snsClient.createTopic(topicRequest);
            return response.topicArn();

        } catch (SnsException 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 .
  - CreateQueue
  - CreateTopic
  - DeleteMessageBatch
  - DeleteQueue
  - DeleteTopic
  - GetQueueAttributes

- Pubblicare
- ReceiveMessage
- SetQueueAttributes
- Subscribe
- Unsubscribe

Usa la libreria di messaggistica Java di Amazon SQS per lavorare con l'interfaccia JMS

Il seguente esempio di codice mostra come utilizzare la libreria di messaggistica Java di Amazon SQS per lavorare con l'interfaccia JMS.

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.

I seguenti esempi funzionano con le code Amazon SQS standard e includono:

- Invio di un messaggio di testo.
- Ricezione di messaggi in modo sincrono.
- Ricezione di messaggi in modo asincrono.
- Ricezione di messaggi utilizzando la modalità CLIENT_CONSENSOR.
- Ricezione di messaggi utilizzando la modalità UNORDERED_ENCOGNOLE.
- Usare Spring per iniettare dipendenze.
- Una classe di utilità che fornisce metodi comuni utilizzati dagli altri esempi.

Per ulteriori informazioni sull'uso di JMS con Amazon SQS, consulta la Amazon SQS Developer Guide.

Invio di un messaggio di testo.

```
    /**
     * This method establishes a connection to a standard Amazon SQS queue using the
Amazon SQS
```

```
    * Java Messaging Library and sends text messages to it. It uses JMS (Java
Message Service) API
    * with automatic acknowledgment mode to ensure reliable message delivery, and
automatically
    * manages all messaging resources.
    *
    * @throws JMSException If there is a problem connecting to or sending messages
to the queue
    */
   public static void doSendTextMessage() throws JMSException {
       // Create a connection factory.
       SQSConnectionFactory connectionFactory = new SQSConnectionFactory(
               new ProviderConfiguration(),
               SqsClient.create()
       );

       // Create the connection in a try-with-resources statement so that it's
closed automatically.
       try (SQSConnection connection = connectionFactory.createConnection()) {

           // Create the queue if needed.
           SqsJmsExampleUtils.ensureQueueExists(connection, QUEUE_NAME,
SqsJmsExampleUtils.QUEUE_VISIBILITY_TIMEOUT);

           // Create a session that uses the JMS auto-acknowledge mode.
           Session session = connection.createSession(false,
Session.AUTO_ACKNOWLEDGE);
           MessageProducer producer =
session.createProducer(session.createQueue(QUEUE_NAME));

           createAndSendMessages(session, producer);
       } // The connection closes automatically. This also closes the session.
       LOGGER.info("Connection closed");
   }

   /**
    * This method reads text input from the keyboard and sends each line as a
separate message
    * to a standard Amazon SQS queue using the Amazon SQS Java Messaging Library.
It continues
    * to accept input until the user enters an empty line, using JMS (Java Message
Service) API to
    * handle the message delivery.
    *
```

```
    * @param session The JMS session used to create messages
    * @param producer The JMS message producer used to send messages to the queue
    */
   private static void createAndSendMessages(Session session, MessageProducer
producer) {
       BufferedReader inputReader = new BufferedReader(
               new InputStreamReader(System.in, Charset.defaultCharset()));

       try {
           String input;
           while (true) {
               LOGGER.info("Enter message to send (leave empty to exit): ");
               input = inputReader.readLine();
               if (input == null || input.isEmpty()) break;

               TextMessage message = session.createTextMessage(input);
               producer.send(message);
               LOGGER.info("Send message {}", message.getJMSMessageID());
           }
       } catch (EOFException e) {
           // Just return on EOF
       } catch (IOException e) {
           LOGGER.error("Failed reading input: {}", e.getMessage(), e);
       } catch (JMSException e) {
           LOGGER.error("Failed sending message: {}", e.getMessage(), e);
       }
   }
```

Ricezione di messaggi in modo sincrono.

```
   /**
    * This method receives messages from a standard Amazon SQS queue using the
Amazon SQS Java
    * Messaging Library. It creates a connection to the queue using JMS (Java
Message Service),
    * waits for messages to arrive, and processes them one at a time. The method
handles all
    * necessary setup and cleanup of messaging resources.
    *
    * @throws JMSException If there is a problem connecting to or receiving
messages from the queue
    */
```

```
   public static void doReceiveMessageSync() throws JMSException {
       // Create a connection factory.
       SQSConnectionFactory connectionFactory = new SQSConnectionFactory(
               new ProviderConfiguration(),
               SqsClient.create()
       );

       // Create a connection.
       try (SQSConnection connection = connectionFactory.createConnection() ) {

           // Create the queue if needed.
           SqsJmsExampleUtils.ensureQueueExists(connection, QUEUE_NAME,
SqsJmsExampleUtils.QUEUE_VISIBILITY_TIMEOUT);

           // Create a session.
           Session session = connection.createSession(false,
Session.CLIENT_ACKNOWLEDGE);
           MessageConsumer consumer =
session.createConsumer(session.createQueue(QUEUE_NAME));

           connection.start();

           receiveMessages(consumer);
       }  // The connection closes automatically. This also closes the session.
       LOGGER.info("Connection closed");
   }

   /**
    * This method continuously checks for new messages from a standard Amazon SQS
queue using
    * the Amazon SQS Java Messaging Library. It waits up to 20 seconds for each
message, processes
    * it using JMS (Java Message Service), and confirms receipt. The method stops
checking for
    * messages after 20 seconds of no activity.
    *
    * @param consumer The JMS message consumer that receives messages from the
queue
    */
   private static void receiveMessages(MessageConsumer consumer) {
       try {
           while (true) {
               LOGGER.info("Waiting for messages...");
               // Wait 1 minute for a message
```

```
                Message message =
consumer.receive(Duration.ofSeconds(20).toMillis());
                if (message == null) {
                    LOGGER.info("Shutting down after 20 seconds of silence.");
                    break;
                }
                SqsJmsExampleUtils.handleMessage(message);
                message.acknowledge();
                LOGGER.info("Acknowledged message {}", message.getJMSMessageID());
            }
        } catch (JMSException e) {
            LOGGER.error("Error receiving from SQS: {}", e.getMessage(), e);
        }
    }
```

Ricezione di messaggi in modo asincrono.

```
    /**
     * This method sets up automatic message handling for a standard Amazon SQS
queue using the
     * Amazon SQS Java Messaging Library. It creates a listener that processes
messages as soon
     * as they arrive using JMS (Java Message Service), runs for 5 seconds, then
cleans up all
     * messaging resources.
     *
     * @throws JMSException If there is a problem connecting to or receiving
messages from the queue
     */
    public static void doReceiveMessageAsync() throws JMSException {
        // Create a connection factory.
        SQSConnectionFactory connectionFactory = new SQSConnectionFactory(
                new ProviderConfiguration(),
                SqsClient.create()
        );

        // Create a connection.
        try (SQSConnection connection = connectionFactory.createConnection() ) {

            // Create the queue if needed.
            SqsJmsExampleUtils.ensureQueueExists(connection, QUEUE_NAME,
SqsJmsExampleUtils.QUEUE_VISIBILITY_TIMEOUT);
```

```
            // Create a session.
            Session session = connection.createSession(false,
Session.CLIENT_ACKNOWLEDGE);

            try {
                // Create a consumer for the queue.
                MessageConsumer consumer =
session.createConsumer(session.createQueue(QUEUE_NAME));
                // Provide an implementation of the MessageListener interface, which
has a single 'onMessage' method.
                // We use a lambda expression for the implementation.
                consumer.setMessageListener(message -> {
                    try {
                        SqsJmsExampleUtils.handleMessage(message);
                        message.acknowledge();
                    } catch (JMSException e) {
                        LOGGER.error("Error processing message: {}",
e.getMessage());
                    }
                });
                // Start receiving incoming messages.
                connection.start();
                LOGGER.info("Waiting for messages...");
            } catch (JMSException e) {
                throw new RuntimeException(e);
            }
            try {
                Thread.sleep(5000);
            } catch (InterruptedException e) {
                throw new RuntimeException(e);
            }
        }  // The connection closes automatically. This also closes the session.
        LOGGER.info( "Connection closed" );
    }
```

Ricezione di messaggi utilizzando la modalità CLIENT_CONSENSOR.

```
    /**
     * This method demonstrates how message acknowledgment affects message
processing in a standard
```

```
    * Amazon SQS queue using the Amazon SQS Java Messaging Library. It sends
messages to the queue,
    * then shows how JMS (Java Message Service) client acknowledgment mode handles
both explicit
    * and implicit message confirmations, including how acknowledging one message
can automatically
    * acknowledge previous messages.
    *
    * @throws JMSException If there is a problem with the messaging operations
    */
   public static void doReceiveMessagesSyncClientAcknowledge() throws JMSException
{
       // Create a connection factory.
       SQSConnectionFactory connectionFactory = new SQSConnectionFactory(
               new ProviderConfiguration(),
               SqsClient.create()
       );

       // Create the connection in a try-with-resources statement so that it's
closed automatically.
       try (SQSConnection connection = connectionFactory.createConnection() ) {

           // Create the queue if needed.
           SqsJmsExampleUtils.ensureQueueExists(connection, QUEUE_NAME,
TIME_OUT_SECONDS);

           // Create a session with client acknowledge mode.
           Session session = connection.createSession(false,
Session.CLIENT_ACKNOWLEDGE);

           // Create a producer and consumer.
           MessageProducer producer =
session.createProducer(session.createQueue(QUEUE_NAME));
           MessageConsumer consumer =
session.createConsumer(session.createQueue(QUEUE_NAME));

           // Open the connection.
           connection.start();

           // Send two text messages.
           sendMessage(producer, session, "Message 1");
           sendMessage(producer, session, "Message 2");

           // Receive a message and don't acknowledge it.
```

```
            receiveMessage(consumer, false);

            // Receive another message and acknowledge it.
            receiveMessage(consumer, true);

            // Wait for the visibility time out, so that unacknowledged messages
reappear in the queue,
            LOGGER.info("Waiting for visibility timeout...");
            try {
                Thread.sleep(TIME_OUT_MILLIS);
            } catch (InterruptedException e) {
                LOGGER.error("Interrupted while waiting for visibility timeout", e);
                Thread.currentThread().interrupt();
                throw new RuntimeException("Processing interrupted", e);
            }

            /*  We will attempt to receive another message, but none will be
available. This is because in
                CLIENT_ACKNOWLEDGE mode, when we acknowledged the second message,
all previous messages were
                automatically acknowledged as well. Therefore, although we never
directly acknowledged the first
                message, it was implicitly acknowledged when we confirmed the second
one. */
            receiveMessage(consumer, true);
        } // The connection closes automatically. This also closes the session.
        LOGGER.info("Connection closed.");

    }


    /**
     * Sends a text message using the specified JMS MessageProducer and Session.
     *
     * @param producer    The JMS MessageProducer used to send the message
     * @param session     The JMS Session used to create the text message
     * @param messageText The text content to be sent in the message
     * @throws JMSException If there is an error creating or sending the message
     */
    private static void sendMessage(MessageProducer producer, Session session,
String messageText) throws JMSException {
        // Create a text message and send it.
        producer.send(session.createTextMessage(messageText));
    }
```

```
   /**
    * Receives and processes a message from a JMS queue using the specified
consumer.
    * The method waits for a message until the configured timeout period is
reached.
    * If a message is received, it is logged and optionally acknowledged based on
the
    * acknowledge parameter.
    *
    * @param consumer    The JMS MessageConsumer used to receive messages from the
queue
    * @param acknowledge Boolean flag indicating whether to acknowledge the
message.
    *                    If true, the message will be acknowledged after processing
    * @throws JMSException If there is an error receiving, processing, or
acknowledging the message
    */
   private static void receiveMessage(MessageConsumer consumer, boolean
acknowledge) throws JMSException {
       // Receive a message.
       Message message = consumer.receive(TIME_OUT_MILLIS);

       if (message == null) {
           LOGGER.info("Queue is empty!");
       } else {
           // Since this queue has only text messages, cast the message object and
print the text.
           LOGGER.info("Received: {}    Acknowledged: {}", ((TextMessage)
message).getText(), acknowledge);

           // Acknowledge the message if asked.
           if (acknowledge) message.acknowledge();
       }
   }
```

Ricezione di messaggi utilizzando la modalità UNORDERED_ENCOGNOLE.

```
   /**
    * Demonstrates message acknowledgment behavior in UNORDERED_ACKNOWLEDGE mode
with Amazon SQS JMS.
```

```
    * In this mode, each message must be explicitly acknowledged regardless of
receive order.
    * Unacknowledged messages return to the queue after the visibility timeout
expires,
    * unlike CLIENT_ACKNOWLEDGE mode where acknowledging one message acknowledges
all previous messages.
    *
    * @throws JMSException         If a JMS-related error occurs during message
operations
    */
   public static void doReceiveMessagesUnorderedAcknowledge() throws JMSException {
       // Create a connection factory.
       SQSConnectionFactory connectionFactory = new SQSConnectionFactory(
               new ProviderConfiguration(),
               SqsClient.create()
       );

       // Create the connection in a try-with-resources statement so that it's
closed automatically.
       try( SQSConnection connection = connectionFactory.createConnection() ) {

           // Create the queue if needed.
           SqsJmsExampleUtils.ensureQueueExists(connection, QUEUE_NAME,
TIME_OUT_SECONDS);

           // Create a session with unordered acknowledge mode.
           Session session = connection.createSession(false,
SQSSession.UNORDERED_ACKNOWLEDGE);

           // Create the producer and consumer.
           MessageProducer producer =
session.createProducer(session.createQueue(QUEUE_NAME));
           MessageConsumer consumer =
session.createConsumer(session.createQueue(QUEUE_NAME));

           // Open a connection.
           connection.start();

           // Send two text messages.
           sendMessage(producer, session, "Message 1");
           sendMessage(producer, session, "Message 2");

           // Receive a message and don't acknowledge it.
           receiveMessage(consumer, false);
```

```
            // Receive another message and acknowledge it.
            receiveMessage(consumer, true);

            // Wait for the visibility time out, so that unacknowledged messages
reappear in the queue.
            LOGGER.info("Waiting for visibility timeout...");
            try {
                Thread.sleep(TIME_OUT_MILLIS);
            } catch (InterruptedException e) {
                LOGGER.error("Interrupted while waiting for visibility timeout", e);
                Thread.currentThread().interrupt();
                throw new RuntimeException("Processing interrupted", e);
            }

            /*  We will attempt to receive another message, and we'll get the first
message again. This occurs
                because in UNORDERED_ACKNOWLEDGE mode, each message requires its own
separate acknowledgment.
                Since we only acknowledged the second message, the first message
remains in the queue for
                redelivery. */
            receiveMessage(consumer, true);

            LOGGER.info("Connection closed.");
        } // The connection closes automatically. This also closes the session.
    }

    /**
     * Sends a text message to an Amazon SQS queue using JMS.
     *
     * @param producer    The JMS MessageProducer for the queue
     * @param session     The JMS Session for message creation
     * @param messageText The message content
     * @throws JMSException If message creation or sending fails
     */
    private static void sendMessage(MessageProducer producer, Session session,
String messageText) throws JMSException {
        // Create a text message and send it.
        producer.send(session.createTextMessage(messageText));
    }
    /**
     * Synchronously receives a message from an Amazon SQS queue using the JMS API
     * with an acknowledgment parameter.
```

```
     *
     * @param consumer    The JMS MessageConsumer for the queue
     * @param acknowledge If true, acknowledges the message after receipt
     * @throws JMSException If message reception or acknowledgment fails
     */
    private static void receiveMessage(MessageConsumer consumer, boolean
 acknowledge) throws JMSException {
        // Receive a message.
        Message message = consumer.receive(TIME_OUT_MILLIS);

        if (message == null) {
            LOGGER.info("Queue is empty!");
        } else {
            // Since this queue has only text messages, cast the message object and
 print the text.
            LOGGER.info("Received: {}    Acknowledged: {}", ((TextMessage)
 message).getText(), acknowledge);

            // Acknowledge the message if asked.
            if (acknowledge) message.acknowledge();
        }
    }
```

Usare Spring per iniettare dipendenze.

```
package com.example.sqs.jms.spring;

import com.amazon.sqs.javamessaging.SQSConnection;
import com.example.sqs.jms.SqsJmsExampleUtils;
import jakarta.jms.*;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.NoSuchBeanDefinitionException;
import org.springframework.context.support.FileSystemXmlApplicationContext;

import java.io.File;
import java.net.URL;
import java.util.concurrent.TimeUnit;

/**
 * Demonstrates how to send and receive messages using the Amazon SQS Java Messaging
 Library
```

```
 * with Spring Framework integration. This example connects to a standard Amazon SQS
 message
 * queue using Spring's dependency injection to configure the connection and
 messaging components.
 * The application uses the JMS (Java Message Service) API to handle message
 operations.
 */
public class SpringExample {
    private static final Integer POLLING_SECONDS = 15;
    private static final String SPRING_XML_CONFIG_FILE =
 "SpringExampleConfiguration.xml.txt";
    private static final Logger LOGGER =
 LoggerFactory.getLogger(SpringExample.class);

    /**
     * Demonstrates sending and receiving messages through a standard Amazon SQS
 message queue
     * using Spring Framework configuration. This method loads connection settings
 from an XML file,
     * establishes a messaging session using the Amazon SQS Java Messaging Library,
 and processes
     * messages using JMS (Java Message Service) operations. If the queue doesn't
 exist, it will
     * be created automatically.
     *
     * @param args Command line arguments (not used)
     */
    public static void main(String[] args) {

        URL resource =
 SpringExample.class.getClassLoader().getResource(SPRING_XML_CONFIG_FILE);
        File springFile = new File(resource.getFile());
        if (!springFile.exists() || !springFile.canRead()) {
            LOGGER.error("File " + SPRING_XML_CONFIG_FILE + " doesn't exist or isn't
 readable.");
            System.exit(1);
        }

        try (FileSystemXmlApplicationContext context =
                     new FileSystemXmlApplicationContext("file://" +
 springFile.getAbsolutePath())) {

            Connection connection;
            try {
```

```
                connection = context.getBean(Connection.class);
            } catch (NoSuchBeanDefinitionException e) {
                LOGGER.error("Can't find the JMS connection to use: " +
e.getMessage(), e);
                System.exit(2);
                return;
            }

            String queueName;
            try {
                queueName = context.getBean("queueName", String.class);
            } catch (NoSuchBeanDefinitionException e) {
                LOGGER.error("Can't find the name of the queue to use: " +
e.getMessage(), e);
                System.exit(3);
                return;
            }
            try {
                if (connection instanceof SQSConnection) {
                    SqsJmsExampleUtils.ensureQueueExists((SQSConnection) connection,
queueName, SqsJmsExampleUtils.QUEUE_VISIBILITY_TIMEOUT);
                }
                // Create the JMS session.
                Session session = connection.createSession(false,
Session.CLIENT_ACKNOWLEDGE);

                SqsJmsExampleUtils.sendTextMessage(session, queueName);
                MessageConsumer consumer =
session.createConsumer(session.createQueue(queueName));

                receiveMessages(consumer);
            } catch (JMSException e) {
                LOGGER.error(e.getMessage(), e);
                throw new RuntimeException(e);
            }
        }   // Spring context autocloses. Managed Spring beans that implement
AutoClosable, such as the
        // 'connection' bean, are also closed.
        LOGGER.info("Context closed");
    }

    /**
     * Continuously checks for and processes messages from a standard Amazon SQS
message queue
```

```
     * using the Amazon SQS Java Messaging Library underlying the JMS API. This
 method waits for incoming messages,
     * processes them when they arrive, and acknowledges their receipt using JMS
 (Java Message
     * Service) operations. The method will stop checking for messages after 15
 seconds of
     * inactivity.
     *
     * @param consumer The JMS message consumer used to receive messages from the
 queue
     */
    private static void receiveMessages(MessageConsumer consumer) {
        try {
            while (true) {
                LOGGER.info("Waiting for messages...");
                // Wait 15 seconds for a message.
                Message message =
 consumer.receive(TimeUnit.SECONDS.toMillis(POLLING_SECONDS));
                if (message == null) {
                    LOGGER.info("Shutting down after {} seconds of silence.",
 POLLING_SECONDS);
                    break;
                }
                SqsJmsExampleUtils.handleMessage(message);
                message.acknowledge();
                LOGGER.info("Message acknowledged.");
            }
        } catch (JMSException e) {
            LOGGER.error("Error receiving from SQS.", e);
        }
    }
}
```

Definizioni di Spring Beans.

```
<?xml version="1.0" encoding="UTF-8"?>
<beans
        xmlns="http://www.springframework.org/schema/beans"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="
http://www.springframework.org/schema/beans http://www.springframework.org/schema/
beans/spring-beans-3.0.xsd
```

```
        ">
    <!-- Define the AWS Region -->
    <bean id="region" class="software.amazon.awssdk.regions.Region" factory-
method="of">
        <constructor-arg value="us-east-1"/>
    </bean>

    <bean id="credentialsProviderBean"
 class="software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider"
          factory-method="create"/>

    <bean id="clientBuilder" class="software.amazon.awssdk.services.sqs.SqsClient"
 factory-method="builder"/>

    <bean id="regionSetClientBuilder" factory-bean="clientBuilder" factory-
method="region">
        <constructor-arg ref="region"/>
    </bean>

    <!-- Configure the Builder with Credentials Provider -->
    <bean id="sqsClient" factory-bean="regionSetClientBuilder" factory-
method="credentialsProvider">
        <constructor-arg ref="credentialsProviderBean"/>
    </bean>

    <bean id="providerConfiguration"
 class="com.amazon.sqs.javamessaging.ProviderConfiguration">
        <property name="numberOfMessagesToPrefetch" value="5"/>
    </bean>

    <bean id="connectionFactory"
 class="com.amazon.sqs.javamessaging.SQSConnectionFactory">
        <constructor-arg ref="providerConfiguration"/>
        <constructor-arg ref="clientBuilder"/>
    </bean>

    <bean id="connection"
          factory-bean="connectionFactory"
          factory-method="createConnection"
          init-method="start"
          destroy-method="close"/>

    <bean id="queueName" class="java.lang.String">
        <constructor-arg value="SQSJMSClientExampleQueue"/>
```

```
    </bean>
</beans>
```

Una classe di utilità che fornisce metodi comuni usati dagli altri esempi.

```
package com.example.sqs.jms;

import com.amazon.sqs.javamessaging.AmazonSQSMessagingClientWrapper;
import com.amazon.sqs.javamessaging.ProviderConfiguration;
import com.amazon.sqs.javamessaging.SQSConnection;
import com.amazon.sqs.javamessaging.SQSConnectionFactory;
import jakarta.jms.*;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.exception.SdkException;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.CreateQueueRequest;
import software.amazon.awssdk.services.sqs.model.QueueAttributeName;

import java.time.Duration;
import java.util.Base64;
import java.util.Map;

/**
 * This utility class provides helper methods for working with Amazon Simple Queue
 Service (Amazon SQS)
 * through the Java Message Service (JMS) interface. It contains common operations
 for managing message
 * queues and handling message delivery.
 */
public class SqsJmsExampleUtils {
    private static final Logger LOGGER =
 LoggerFactory.getLogger(SqsJmsExampleUtils.class);
    public static final Long QUEUE_VISIBILITY_TIMEOUT = 5L;

    /**
     * This method verifies that a message queue exists and creates it if necessary.
 The method checks for
     * an existing queue first to optimize performance.
     *
     * @param connection The active connection to the messaging service
```

```
    * @param queueName The name of the queue to verify or create
    * @param visibilityTimeout The duration in seconds that messages will be hidden
after being received
    * @throws JMSException If there is an error accessing or creating the queue
    */
   public static void ensureQueueExists(SQSConnection connection, String queueName,
Long visibilityTimeout) throws JMSException {
       AmazonSQSMessagingClientWrapper client =
connection.getWrappedAmazonSQSClient();

      /* In most cases, you can do this with just a 'createQueue' call, but
'getQueueUrl'
      (called by 'queueExists') is a faster operation for the common case where the
queue
      already exists. Also, many users and roles have permission to call
'getQueueUrl'
      but don't have permission to call 'createQueue'.
      */
       if( !client.queueExists(queueName) ) {
           CreateQueueRequest createQueueRequest = CreateQueueRequest.builder()
                   .queueName(queueName)
                   .attributes(Map.of(QueueAttributeName.VISIBILITY_TIMEOUT,
String.valueOf(visibilityTimeout)))
                   .build();
           client.createQueue( createQueueRequest );
       }
   }

   /**
    * This method sends a simple text message to a specified message queue. It
handles all necessary
    * setup for the message delivery process.
    *
    * @param session The active messaging session used to create and send the
message
    * @param queueName The name of the queue where the message will be sent
    */
   public static void sendTextMessage(Session session, String queueName) {
       // Rest of implementation...

       try {
           MessageProducer producer =
session.createProducer( session.createQueue( queueName) );
           Message message = session.createTextMessage("Hello world!");
```

```
            producer.send(message);
        } catch (JMSException e) {
            LOGGER.error( "Error receiving from SQS", e );
        }
    }

    /**
     * This method processes incoming messages and logs their content based on the
message type.
     * It supports text messages, binary data, and Java objects.
     *
     * @param message The message to be processed and logged
     * @throws JMSException If there is an error reading the message content
     */
    public static void handleMessage(Message message) throws JMSException {
        // Rest of implementation...
        LOGGER.info( "Got message {}", message.getJMSMessageID() );
        LOGGER.info( "Content: ");
        if(message instanceof TextMessage txtMessage) {
            LOGGER.info( "\t{}", txtMessage.getText() );
        } else if(message instanceof BytesMessage byteMessage){
            // Assume the length fits in an int - SQS only supports sizes up to 256k
so that
            // should be true
            byte[] bytes = new byte[(int)byteMessage.getBodyLength()];
            byteMessage.readBytes(bytes);
            LOGGER.info( "\t{}", Base64.getEncoder().encodeToString( bytes ) );
        } else if( message instanceof ObjectMessage) {
            ObjectMessage objMessage = (ObjectMessage) message;
            LOGGER.info( "\t{}", objMessage.getObject() );
        }
    }

    /**
     * This method sets up automatic message processing for a specified queue. It
creates a listener
     * that will receive and handle incoming messages without blocking the main
program.
     *
     * @param session The active messaging session
     * @param queueName The name of the queue to monitor
     * @param connection The active connection to the messaging service
     */
```

```
   public static void receiveMessagesAsync(Session session, String queueName,
Connection connection) {
       // Rest of implementation...
       try {
           // Create a consumer for the queue.
           MessageConsumer consumer =
session.createConsumer(session.createQueue(queueName));
           // Provide an implementation of the MessageListener interface, which has
a single 'onMessage' method.
           // We use a lambda expression for the implementation.
           consumer.setMessageListener(message -> {
               try {
                   SqsJmsExampleUtils.handleMessage(message);
                   message.acknowledge();
               } catch (JMSException e) {
                   LOGGER.error("Error processing message: {}", e.getMessage());
               }
           });
           // Start receiving incoming messages.
           connection.start();
       } catch (JMSException e) {
           throw new RuntimeException(e);
       }
       try {
           Thread.sleep(2000);
       } catch (InterruptedException e) {
           throw new RuntimeException(e);
       }
   }


   /**
    * This method performs cleanup operations after message processing is complete.
It receives
    * any messages in the specified queue, removes the message queue and closes all
    * active connections to prevent resource leaks.
    *
    * @param queueName The name of the queue to be removed
    * @param visibilityTimeout The duration in seconds that messages are hidden
after being received
    * @throws JMSException If there is an error during the cleanup process
    */
   public static void cleanUpExample(String queueName, Long visibilityTimeout)
throws JMSException {
```

```
        LOGGER.info("Performing cleanup.");

        SQSConnectionFactory connectionFactory = new SQSConnectionFactory(
                new ProviderConfiguration(),
                SqsClient.create()
        );

        try (SQSConnection connection = connectionFactory.createConnection() ) {
            ensureQueueExists(connection, queueName, visibilityTimeout);
            Session session = connection.createSession(false,
Session.AUTO_ACKNOWLEDGE);

            receiveMessagesAsync(session, queueName, connection);

            SqsClient sqsClient =
connection.getWrappedAmazonSQSClient().getAmazonSQSClient();
            try {
                String queueUrl = sqsClient.getQueueUrl(b ->
b.queueName(queueName)).queueUrl();
                sqsClient.deleteQueue(b -> b.queueUrl(queueUrl));
                LOGGER.info("Queue deleted: {}", queueUrl);
            } catch (SdkException e) {
                LOGGER.error("Error during SQS operations: ", e);
            }
        }
        LOGGER.info("Clean up: Connection closed");
    }

    /**
     * This method creates a background task that sends multiple messages to a
specified queue
     * after waiting for a set time period. The task operates independently to
ensure efficient
     * message processing without interrupting other operations.
     *
     * @param queueName The name of the queue where messages will be sent
     * @param secondsToWait The number of seconds to wait before sending messages
     * @param numMessages The number of messages to send
     * @param visibilityTimeout The duration in seconds that messages remain hidden
after being received
     * @return A task that can be executed to send the messages
     */
    public static Runnable sendAMessageAsync(String queueName, Long secondsToWait,
Integer numMessages, Long visibilityTimeout) {
```

```
        return () -> {
            try {
                Thread.sleep(Duration.ofSeconds(secondsToWait).toMillis());
            } catch (InterruptedException e) {
                Thread.currentThread().interrupt();
                throw new RuntimeException(e);
            }
            try {
                SQSConnectionFactory connectionFactory = new SQSConnectionFactory(
                        new ProviderConfiguration(),
                        SqsClient.create()
                );
                try (SQSConnection connection =
 connectionFactory.createConnection()) {
                    ensureQueueExists(connection, queueName, visibilityTimeout);
                    Session session = connection.createSession(false,
 Session.CLIENT_ACKNOWLEDGE);
                    for (int i = 1; i <= numMessages; i++) {
                        MessageProducer producer =
 session.createProducer(session.createQueue(queueName));
                        producer.send(session.createTextMessage("Hello World " + i +
 "!"));
                    }
                }
            } catch (JMSException e) {
                LOGGER.error(e.getMessage(), e);
                throw new RuntimeException(e);
            }
        };
    }
}
```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella Documentazione di riferimento delle API AWS SDK for Java 2.x .
  - CreateQueue
  - DeleteQueue

Lavora con i tag di coda

Il seguente esempio di codice mostra come eseguire operazioni di tagging con Amazon SQS.

SDK per Java 2.x

> **Note**
>
> C'è 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 crea tag per una coda, elenca i tag e rimuove un tag.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.ListQueueTagsResponse;
import software.amazon.awssdk.services.sqs.model.QueueDoesNotExistException;
import software.amazon.awssdk.services.sqs.model.SqsException;

import java.util.Map;
import java.util.UUID;

/**
 * Before running this Java V2 code example, set up your development environment,
 including your credentials. For more
 * information, see the <a href="https://docs.aws.amazon.com/sdk-for-java/latest/
developer-guide/get-started.html">AWS
 * SDK for Java Developer Guide</a>.
 */
public class TagExamples {
    static final SqsClient sqsClient = SqsClient.create();
    static final String queueName = "TagExamples-queue-" +
 UUID.randomUUID().toString().replace("-", "").substring(0, 20);
    private static final Logger LOGGER = LoggerFactory.getLogger(TagExamples.class);

    public static void main(String[] args) {
        final String queueUrl;
        try {
            queueUrl = sqsClient.createQueue(b ->
 b.queueName(queueName)).queueUrl();
            LOGGER.info("Queue created. The URL is: {}", queueUrl);
        } catch (RuntimeException e) {
            LOGGER.error("Program ending because queue was not created.");
```

```
            throw new RuntimeException(e);
        }
        try {
            addTags(queueUrl);
            listTags(queueUrl);
            removeTags(queueUrl);
        } catch (RuntimeException e) {
            LOGGER.error("Program ending because of an error in a method.");
        } finally {
            try {
                sqsClient.deleteQueue(b -> b.queueUrl(queueUrl));
                LOGGER.info("Queue successfully deleted. Program ending.");
                sqsClient.close();
            } catch (RuntimeException e) {
                LOGGER.error("Program ending.");
            } finally {
                sqsClient.close();
            }
        }
    }

    /** This method demonstrates how to use a Java Map to a tag a aueue.
     * @param queueUrl The URL of the queue to tag.
     */
    public static void addTags(String queueUrl) {
        // Build a map of the tags.
        final Map<String, String> tagsToAdd = Map.of(
                "Team", "Development",
                "Priority", "Beta",
                "Accounting ID", "456def");

        try {
            // Add tags to the queue using a Consumer<TagQueueRequest.Builder>
parameter.
            sqsClient.tagQueue(b -> b
                    .queueUrl(queueUrl)
                    .tags(tagsToAdd)
            );
        } catch (QueueDoesNotExistException e) {
            LOGGER.error("Queue does not exist: {}", e.getMessage(), e);
            throw new RuntimeException(e);
        }
    }
```

```
    /** This method demonstrates how to view the tags for a queue.
     * @param queueUrl The URL of the queue whose tags you want to list.
     */
    public static void listTags(String queueUrl) {
        ListQueueTagsResponse response;
        try {
            // Call the listQueueTags method with a
 Consumer<ListQueueTagsRequest.Builder> parameter that creates a
 ListQueueTagsRequest.
            response = sqsClient.listQueueTags(b -> b
                    .queueUrl(queueUrl));
        } catch (SqsException e) {
            LOGGER.error("Exception thrown: {}", e.getMessage(), e);
            throw new RuntimeException(e);
        }

        // Log the tags.
        response.tags()
                .forEach((k, v) ->
                        LOGGER.info("Key: {} -> Value: {}", k, v));
    }

    /**
     * This method demonstrates how to remove tags from a queue.
     * @param queueUrl The URL of the queue whose tags you want to remove.
     */
    public static void removeTags(String queueUrl) {
        try {
            // Call the untagQueue method with a Consumer<UntagQueueRequest.Builder>
 parameter.
            sqsClient.untagQueue(b -> b
                    .queueUrl(queueUrl)
                    .tagKeys("Accounting ID") // Remove a single tag.
            );
        } catch (SqsException e) {
            LOGGER.error("Exception thrown: {}", e.getMessage(), e);
            throw new RuntimeException(e);
        }
    }
}
```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella Documentazione di riferimento delle API AWS SDK for Java 2.x .
  - ListQueueTags
  - TagQueue
  - UntagQueue

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
            } catch (Exception e) {
                //Add failed message identifier to the batchItemFailures list
                batchItemFailures.add(new
 SQSBatchResponse.BatchItemFailure(message.getMessageId()));
            }
        }
        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 nozioni di base sono esempi di codice che mostrano come eseguire le operazioni essenziali all'interno di un servizio.

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.

Gli scenari sono esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio o combinate con altri Servizi AWS.

Ogni esempio include un collegamento al codice sorgente completo, in cui 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, ListStateMachinesconsulta AWS SDK for Java 2.x API Reference.

Argomenti

- Nozioni di base
- Azioni
- Scenari

Nozioni di base

Informazioni di base

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.
 * <p>
 * https://github.com/awsdocs/aws-doc-sdk-examples/tree/main/resources/sample_files
 * <p>
 * To run this code example, place the chat_sfn_state_machine.json file into
 * your project's resources folder.
 * <p>
 * Also, set up your development environment, including your credentials.
 * <p>
 * For information, see this documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 * <p>
 * This Java code example performs the following tasks:
 * <p>
 * 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.
                jsonFile - The location of the chat_sfn_state_machine.json file. You
can located it in resources/sample_files.
            """;

        if (args.length != 4) {
            System.out.println(usage);
            System.exit(1);
        }

        String roleName = args[0];
        String activityName = args[1];
        String stateMachineName = args[2];
        String jsonFile = args[3];
        String polJSON = """
            {
                "Version": "2012-10-17",
                "Statement": [
                    {
                        "Sid": "",
                        "Effect": "Allow",
                        "Principal": {
                            "Service": "states.amazonaws.com"
                        },
                        "Action": "sts:AssumeRole"
                    }
                ]
            }
            """;


        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);

        // Read the file using FileInputStream
        FileInputStream inputStream = new FileInputStream(jsonFile);
        ObjectMapper mapper = new ObjectMapper();
        JsonNode jsonNode = mapper.readValue(inputStream, 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

Azioni

`CreateActivity`

Il seguente esempio di codice mostra come usare`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, consulta la CreateActivitysezione 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, consulta la CreateStateMachinesezione 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, consulta la DeleteActivitysezione 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, consulta la DeleteStateMachinesezione 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, consulta la DescribeExecutionsezione 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, consulta la DescribeStateMachinesezione 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;
   }
```

- Per i dettagli sull'API, consulta la GetActivityTasksezione 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, consulta la ListActivitiessezione 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, consulta la [ListExecutions](sezione 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, consulta la ListStateMachinessezione 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, consulta la SendTaskSuccesssezione 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, consulta la StartExecutionsezione AWS SDK for Java 2.x API Reference.

Scenari

Utilizzo di Step Functions per richiamare le funzioni Lambda

Il seguente esempio di codice mostra come creare una macchina a AWS Step Functions stati che richiama AWS Lambda funzioni in sequenza.

SDK per Java 2.x

Mostra come creare un flusso di lavoro AWS senza server 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

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.

Ogni esempio include un collegamento al codice sorgente completo, in cui 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, consulta la AssumeRolesezione AWS SDK for Java 2.x API Reference.

Supporto 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 Supporto.

Le nozioni di base sono esempi di codice che mostrano come eseguire le operazioni essenziali all'interno di un servizio.

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.

Ogni esempio include un collegamento al codice sorgente completo, in cui è possibile trovare istruzioni su come configurare ed eseguire il codice nel contesto.

Nozioni di base

Ciao Supporto

L'esempio di codice seguente mostra come iniziare a utilizzare Supporto.

SDK per Java 2.x

> **Note**
>
> C'è 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, consulta la DescribeServicessezione AWS SDK for Java 2.x API Reference.

Argomenti

- Nozioni di base
- Azioni

Nozioni di base

Informazioni di base

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 di più su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

Esegui varie Supporto 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 = "C:\\AWS\\test.txt" ; //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

Azioni

`AddAttachmentsToSet`

Il seguente esempio di codice mostra come usare`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, consulta la AddAttachmentsToSetsezione 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, consulta la AddCommunicationToCasesezione 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, consulta la CreateCasesezione 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, consulta la DescribeAttachmentsezione 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, consulta la DescribeCasessezione 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, consulta la DescribeCommunicationssezione 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, consulta la DescribeServicessezione 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, consulta la DescribeSeverityLevelssezione 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, consulta la ResolveCasesezione AWS SDK for Java 2.x API Reference.

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 nozioni di base sono esempi di codice che mostrano come eseguire le operazioni essenziali all'interno di un servizio.

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.

Ogni esempio include un collegamento al codice sorgente completo, in cui è possibile trovare istruzioni su come configurare ed eseguire il codice nel contesto.

Nozioni di base

Salve Systems Manager

L'esempio di codice seguente mostra come iniziare a utilizzare Systems Manager.

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.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 la ListDocumentssezione AWS SDK for Java 2.x API Reference.

Argomenti

- Nozioni di base
- Azioni

Nozioni di base

Informazioni di base

L'esempio di codice seguente mostra come:

- Crea una finestra di manutenzione.
- Modifica la pianificazione della finestra di manutenzione.
- Creare un documento.
- Invia un comando a un' EC2 istanza specificata.
- Crea un OpsItem.
- Aggiorna e risolvi il OpsItem.
- Eliminare la finestra di manutenzione OpsItem e il documento.

SDK per Java 2.x

> **Note**
>
> C'è 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.ssm.model.DocumentAlreadyExistsException;
import software.amazon.awssdk.services.ssm.model.SsmException;

import java.util.Scanner;
public class SSMScenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    public static void main(String[] args) {
        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);
        SSMActions actions = new SSMActions();
        String documentName;
        String windowName;
        String instanceId = args[0];
        String title = "Disk Space Alert" ;
        String source = "EC2" ;
        String category = "Availability" ;
        String severity = "2" ;

        System.out.println(DASHES);
        System.out.println("""
                Welcome to the AWS Systems Manager SDK Basics scenario.
                This Java program demonstrates how to interact with AWS Systems
Manager using the AWS SDK for Java (v2).
                AWS Systems Manager is the operations hub for your AWS applications
and resources and a secure end-to-end management solution.
                The program's primary functionalities 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 AWS SSM resources.
                Upon completion of the program, all AWS resources are cleaned up.
                Let's get started...

                """);
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println("1. Create an SSM 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 = null;
        try {
            winId = actions.createMaintenanceWindow(windowName);
            waitForInputToContinue(scanner);
            System.out.println("The maintenance window ID is: " + winId);
```

```
        } catch (DocumentAlreadyExistsException e) {
            System.err.println("The SSM maintenance window already exists.
Retrieving existing window ID...");
            String existingWinId = actions.createMaintenanceWindow(windowName);
            System.out.println("Existing window ID: " + existingWinId);
        } catch (SsmException e) {
            System.err.println("SSM error: " + e.getMessage());
            return;
        } catch (RuntimeException e) {
            System.err.println("Unexpected error: " + e.getMessage());
            return;
        }
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println("2. Modify the maintenance window by changing the
schedule");
        waitForInputToContinue(scanner);
        try {
            actions.updateSSMMaintenanceWindow(winId, windowName);
            waitForInputToContinue(scanner);
            System.out.println("The SSM maintenance window was successfully
updated");
        } catch (SsmException e) {
            System.err.println("SSM error: " + e.getMessage());
            return;
        } catch (RuntimeException e) {
            System.err.println("Unexpected error: " + e.getMessage());
            return;
        }
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println("3. Create an SSM document that defines the actions that
Systems Manager performs on your managed nodes.");
        System.out.println("Please enter the document name (default is
ssmdocument):");
        String doc = scanner.nextLine();
        documentName = doc.isEmpty() ? "ssmdocument" : doc;
        try {
            actions.createSSMDoc(documentName);
            waitForInputToContinue(scanner);
            System.out.println("The SSM document was successfully created");
        } catch (DocumentAlreadyExistsException e) {
```

```
            System.err.println("The SSM document already exists. Moving on");
        } catch (SsmException e) {
            System.err.println("SSM error: " + e.getMessage());
            return;
        } catch (RuntimeException e) {
            System.err.println("Unexpected error: " + e.getMessage());
        }
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println("4. Now we are going to run a command on an EC2
instance");
        waitForInputToContinue(scanner);
        String commandId="";
        try {
            commandId = actions.sendSSMCommand(documentName, instanceId);
            waitForInputToContinue(scanner);
            System.out.println("The command was successfully sent. Command ID: " +
commandId);
        } catch (SsmException e) {
            System.err.println("SSM error: " + e.getMessage());
        } catch (InterruptedException e) {
            System.err.println("Thread was interrupted: " + e.getMessage());
        } catch (RuntimeException e) {
            System.err.println("Unexpected error: " + e.getMessage());
        }
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println("5. Lets get the time when the specific command was sent
to the specific managed node");
        waitForInputToContinue(scanner);
        try {
            actions.displayCommands(commandId);
            System.out.println("The command invocations were successfully
displayed.");
        } catch (SsmException e) {
            System.err.println("SSM error: " + e.getMessage());
            return;
        } catch (RuntimeException e) {
            System.err.println("Unexpected error: " + e.getMessage());
            return;
        }
        waitForInputToContinue(scanner);
```

```
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("""
             6. Now we will create an SSM OpsItem.
             A SSM OpsItem is a feature provided by Amazon's Systems Manager (SSM)
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.
            """);

        waitForInputToContinue(scanner);
        String opsItemId;
        try {
            opsItemId = actions.createSSMOpsItem(title, source, category, severity);
            System.out.println(opsItemId + " was created");
        } catch (SsmException e) {
            System.err.println("SSM error: " + e.getMessage());
            return;
        } catch (RuntimeException e) {
            System.err.println("Unexpected error: " + e.getMessage());
            return;
        }
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Now we will update the SSM OpsItem "+opsItemId);
        waitForInputToContinue(scanner);
        String description = "An update to "+opsItemId ;
        try {
            actions.updateOpsItem(opsItemId, title, description);
        } catch (SsmException e) {
            System.err.println("SSM error: " + e.getMessage());
            return;
        } catch (RuntimeException e) {
            System.err.println("Unexpected error: " + e.getMessage());
```

```
            return;
        }

        System.out.println(DASHES);
        System.out.println("8. Now we will get the status of the SSM OpsItem
"+opsItemId);
        waitForInputToContinue(scanner);
        try {
            actions.describeOpsItems(opsItemId);
        } catch (SsmException e) {
            System.err.println("SSM error: " + e.getMessage());
            return;
        } catch (RuntimeException e) {
            System.err.println("Unexpected error: " + e.getMessage());
            return;
        }

        System.out.println(DASHES);
        System.out.println("9. Now we will resolve the SSM OpsItem "+opsItemId);
        waitForInputToContinue(scanner);
        try {
            actions.resolveOpsItem(opsItemId);
        } catch (SsmException e) {
            System.err.println("SSM error: " + e.getMessage());
            return;
        } catch (RuntimeException e) {
            System.err.println("Unexpected error: " + e.getMessage());
            return;
        }

        System.out.println(DASHES);
        System.out.println("10. Would you like to delete the AWS Systems Manager
resources? (y/n)");
        String delAns = scanner.nextLine().trim();
        if (delAns.equalsIgnoreCase("y")) {
            System.out.println("You selected to delete the resources.");
            waitForInputToContinue(scanner);
            try {
                actions.deleteMaintenanceWindow(winId);
                actions.deleteDoc(documentName);
            } catch (SsmException e) {
                System.err.println("SSM error: " + e.getMessage());
                return;
            } catch (RuntimeException e) {
```

```
                System.err.println("Unexpected error: " + e.getMessage());
                return;
            }
        } else {
            System.out.println("The AWS Systems Manager resources will not be
 deleted");
        }
        System.out.println(DASHES);

        System.out.println("This concludes the AWS Systems Manager SDK Basics
 scenario.");
        System.out.println(DASHES);
    }

    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.");
            }
        }
    }
}
```

Una classe wrapper per i metodi SDK di Systems Manager.

```
public class SSMActions {

    private static SsmAsyncClient ssmAsyncClient;

    private static SsmAsyncClient getAsyncClient() {
        if (ssmAsyncClient == null) {
            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                    .maxConcurrency(100)
```

```
                    .connectionTimeout(Duration.ofSeconds(60))
                    .readTimeout(Duration.ofSeconds(60))
                    .writeTimeout(Duration.ofSeconds(60))
                    .build();

            ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
                    .apiCallTimeout(Duration.ofMinutes(2))
                    .apiCallAttemptTimeout(Duration.ofSeconds(90))
                    .retryPolicy(RetryPolicy.builder()
                            .numRetries(3)
                            .build())
                    .build();

            ssmAsyncClient = SsmAsyncClient.builder()
                    .region(Region.US_EAST_1)
                    .httpClient(httpClient)
                    .overrideConfiguration(overrideConfig)
                    .build();
        }
        return ssmAsyncClient;
    }

    /**
     * Deletes an AWS SSM document asynchronously.
     *
     * @param documentName The name of the document to delete.
     * <p>
     * This method initiates an asynchronous request to delete an SSM document.
     * If an exception occurs, it handles the error appropriately.
     */
    public void deleteDoc(String documentName) {
        DeleteDocumentRequest documentRequest = DeleteDocumentRequest.builder()
                .name(documentName)
                .build();

        CompletableFuture<Void> future = CompletableFuture.runAsync(() -> {
            getAsyncClient().deleteDocument(documentRequest)
                    .thenAccept(response -> {
                        System.out.println("The SSM document was successfully
deleted.");
                    })
                    .exceptionally(ex -> {
                        throw new CompletionException(ex);
```

```
                    }).join();
        }).exceptionally(ex -> {
            Throwable cause = (ex instanceof CompletionException) ? ex.getCause() :
ex;
            if (cause instanceof SsmException) {
                throw new RuntimeException("SSM error: " + cause.getMessage(),
cause);
            } else {
                throw new RuntimeException("Unexpected error: " +
cause.getMessage(), cause);
            }
        });

        try {
            future.join();
        } catch (CompletionException ex) {
            throw ex.getCause() instanceof RuntimeException ? (RuntimeException)
ex.getCause() : ex;
        }
    }

    /**
     * Deletes an AWS SSM Maintenance Window asynchronously.
     *
     * @param winId The ID of the Maintenance Window to delete.
     * <p>
     * This method initiates an asynchronous request to delete an SSM Maintenance
Window.
     * If an exception occurs, it handles the error appropriately.
     */
    public void deleteMaintenanceWindow(String winId) {
        DeleteMaintenanceWindowRequest windowRequest =
DeleteMaintenanceWindowRequest.builder()
                .windowId(winId)
                .build();

        CompletableFuture<Void> future = CompletableFuture.runAsync(() -> {
            getAsyncClient().deleteMaintenanceWindow(windowRequest)
                    .thenAccept(response -> {
                        System.out.println("The maintenance window was successfully
deleted.");
                    })
                    .exceptionally(ex -> {
                        throw new CompletionException(ex);
```

```
                    }).join();
        }).exceptionally(ex -> {
            Throwable cause = (ex instanceof CompletionException) ? ex.getCause() :
ex;
            if (cause instanceof SsmException) {
                throw new RuntimeException("SSM error: " + cause.getMessage(),
cause);
            } else {
                throw new RuntimeException("Unexpected error: " +
cause.getMessage(), cause);
            }
        });

        try {
            future.join();
        } catch (CompletionException ex) {
            throw ex.getCause() instanceof RuntimeException ? (RuntimeException)
ex.getCause() : ex;
        }
    }

    /**
     * Resolves an AWS SSM OpsItem asynchronously.
     *
     * @param opsID The ID of the OpsItem to resolve.
     * <p>
     * This method initiates an asynchronous request to resolve an SSM OpsItem.
     * If an exception occurs, it handles the error appropriately.
     */
    public void resolveOpsItem(String opsID) {
        UpdateOpsItemRequest opsItemRequest = UpdateOpsItemRequest.builder()
                .opsItemId(opsID)
                .status(OpsItemStatus.RESOLVED)
                .build();

        CompletableFuture<Void> future = CompletableFuture.runAsync(() -> {
            getAsyncClient().updateOpsItem(opsItemRequest)
                    .thenAccept(response -> {
                        System.out.println("OpsItem resolved successfully.");
                    })
                    .exceptionally(ex -> {
                        throw new CompletionException(ex);
                    }).join();
        }).exceptionally(ex -> {
```

```
            Throwable cause = (ex instanceof CompletionException) ? ex.getCause() :
ex;
            if (cause instanceof SsmException) {
                throw new RuntimeException("SSM error: " + cause.getMessage(),
cause);
            } else {
                throw new RuntimeException("Unexpected error: " +
cause.getMessage(), cause);
            }
        });

        try {
            future.join();
        } catch (CompletionException ex) {
            throw ex.getCause() instanceof RuntimeException ? (RuntimeException)
ex.getCause() : ex;
        }
    }

    /**
     * Describes AWS SSM OpsItems asynchronously.
     *
     * @param key The key to filter OpsItems by (e.g., OPS_ITEM_ID).
     *
     * This method initiates an asynchronous request to describe SSM OpsItems.
     * If the request is successful, it prints the title and status of each OpsItem.
     * If an exception occurs, it handles the error appropriately.
     */
    public void describeOpsItems(String key) {
        OpsItemFilter filter = OpsItemFilter.builder()
                .key(OpsItemFilterKey.OPS_ITEM_ID)
                .values(key)
                .operator(OpsItemFilterOperator.EQUAL)
                .build();

        DescribeOpsItemsRequest itemsRequest = DescribeOpsItemsRequest.builder()
                .maxResults(10)
                .opsItemFilters(filter)
                .build();

        CompletableFuture<Void> future = CompletableFuture.runAsync(() -> {
            getAsyncClient().describeOpsItems(itemsRequest)
                    .thenAccept(itemsResponse -> {
```

```
                        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());
                        }
                    })
                    .exceptionally(ex -> {
                        throw new CompletionException(ex);
                    }).join();
        }).exceptionally(ex -> {
            Throwable cause = (ex instanceof CompletionException) ? ex.getCause() :
ex;
            if (cause instanceof SsmException) {
                throw new RuntimeException("SSM error: " + cause.getMessage(),
cause);
            } else {
                throw new RuntimeException("Unexpected error: " +
cause.getMessage(), cause);
            }
        });

        try {
            future.join();
        } catch (CompletionException ex) {
            throw ex.getCause() instanceof RuntimeException ? (RuntimeException)
ex.getCause() : ex;
        }
    }

    /**
     * Updates the AWS SSM OpsItem asynchronously.
     *
     * @param opsItemId The ID of the OpsItem to update.
     * @param title The new title of the OpsItem.
     * @param description The new description of the OpsItem.
     * <p>
     * This method initiates an asynchronous request to update an SSM OpsItem.
     * If the request is successful, it completes without returning a value.
     * If an exception occurs, it handles the error appropriately.
     */
    public void updateOpsItem(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());

        CompletableFuture<Void> future = getOpsItem(opsItemId).thenCompose(opsItem -
> {
            UpdateOpsItemRequest request = UpdateOpsItemRequest.builder()
                    .opsItemId(opsItemId)
                    .title(title)
                    .operationalData(operationalData)
                    .status(opsItem.statusAsString())
                    .description(description)
                    .build();

            return getAsyncClient().updateOpsItem(request).thenAccept(response -> {
                System.out.println(opsItemId + " updated successfully.");
            }).exceptionally(ex -> {
                throw new CompletionException(ex);
            });
        }).exceptionally(ex -> {
            Throwable cause = (ex instanceof CompletionException) ? ex.getCause() :
 ex;
            if (cause instanceof SsmException) {
                throw new RuntimeException("SSM error: " + cause.getMessage(),
 cause);
            } else {
                throw new RuntimeException("Unexpected error: " +
 cause.getMessage(), cause);
            }
        });

        try {
            future.join();
        } catch (CompletionException ex) {
            throw ex.getCause() instanceof RuntimeException ? (RuntimeException)
 ex.getCause() : ex;
        }
    }


    private static CompletableFuture<OpsItem> getOpsItem(String opsItemId) {
        GetOpsItemRequest request =
 GetOpsItemRequest.builder().opsItemId(opsItemId).build();
```

```
        return
getAsyncClient().getOpsItem(request).thenApply(GetOpsItemResponse::opsItem);
    }

    /**
     * Creates an SSM OpsItem asynchronously.
     *
     * @param title The title of the OpsItem.
     * @param source The source of the OpsItem.
     * @param category The category of the OpsItem.
     * @param severity The severity of the OpsItem.
     * @return The ID of the created OpsItem.
     * <p>
     * This method initiates an asynchronous request to create an SSM OpsItem.
     * If the request is successful, it returns the OpsItem ID.
     * If an exception occurs, it handles the error appropriately.
     */
    public String createSSMOpsItem(String title, String source, String category,
String severity) {
        CreateOpsItemRequest opsItemRequest = CreateOpsItemRequest.builder()
                .description("Created by the SSM Java API")
                .title(title)
                .source(source)
                .category(category)
                .severity(severity)
                .build();

        CompletableFuture<CreateOpsItemResponse> future =
getAsyncClient().createOpsItem(opsItemRequest);

        try {
            CreateOpsItemResponse response = future.join();
            return response.opsItemId();
        } catch (CompletionException e) {
            Throwable cause = e.getCause();
            if (cause instanceof SsmException) {
                throw (SsmException) cause;
            } else {
                throw new RuntimeException(cause);
            }
        }
    }

    /**
```

```
    * Displays the date and time when the specific command was invoked.
    *
    * @param commandId The ID of the command to describe.
    * <p>
    * This method initiates an asynchronous request to list command invocations and
prints the date and time of each command invocation.
    * If an exception occurs, it handles the error appropriately.
    */
   public void displayCommands(String commandId) {
       ListCommandInvocationsRequest commandInvocationsRequest =
ListCommandInvocationsRequest.builder()
               .commandId(commandId)
               .build();

       CompletableFuture<ListCommandInvocationsResponse> future =
getAsyncClient().listCommandInvocations(commandInvocationsRequest);
       future.thenAccept(response -> {
           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()));
           }
       }).exceptionally(ex -> {
           Throwable cause = (ex instanceof CompletionException) ? ex.getCause() :
ex;
           if (cause instanceof SsmException) {
               throw (SsmException) cause;
           } else {
               throw new RuntimeException(cause);
           }
       }).join();
   }

   /**
    * Sends a SSM command to a managed node asynchronously.
    *
    * @param documentName The name of the document to use.
    * @param instanceId The ID of the instance to send the command to.
    * @return The command ID.
    * <p>
    * This method initiates asynchronous requests to send a SSM command to a
managed node.
```

```
    * It waits until the document is active, sends the command, and checks the
command execution status.
    */
   public String sendSSMCommand(String documentName, String instanceId) throws
InterruptedException, SsmException {
       // Before we use Document to send a command - make sure it is active.
       CompletableFuture<Void> documentActiveFuture = CompletableFuture.runAsync(()
-> {
           boolean isDocumentActive = false;
           DescribeDocumentRequest request = DescribeDocumentRequest.builder()
                   .name(documentName)
                   .build();

           while (!isDocumentActive) {
               CompletableFuture<DescribeDocumentResponse> response =
getAsyncClient().describeDocument(request);
               String documentStatus = response.join().document().statusAsString();
               if (documentStatus.equals("Active")) {
                   System.out.println("The SSM document is active and ready to
use.");
                   isDocumentActive = true;
               } else {
                   System.out.println("The SSM document is not active. Status: " +
documentStatus);
                   try {
                       Thread.sleep(5000);
                   } catch (InterruptedException e) {
                       throw new RuntimeException(e);
                   }
               }
           }
       });

       documentActiveFuture.join();

       // Create the SendCommandRequest.
       SendCommandRequest commandRequest = SendCommandRequest.builder()
               .documentName(documentName)
               .instanceIds(instanceId)
               .build();

       // Send the command.
       CompletableFuture<SendCommandResponse> commandFuture =
getAsyncClient().sendCommand(commandRequest);
```

```
        final String[] commandId = {null};

        commandFuture.whenComplete((commandResponse, ex) -> {
            if (commandResponse != null) {
                commandId[0] = commandResponse.command().commandId();
                System.out.println("Command ID: " + commandId[0]);

                // Wait for the command execution to complete.
                GetCommandInvocationRequest invocationRequest =
GetCommandInvocationRequest.builder()
                        .commandId(commandId[0])
                        .instanceId(instanceId)
                        .build();

                try {
                    System.out.println("Wait 5 secs");
                    TimeUnit.SECONDS.sleep(5);

                    // Retrieve the command execution details.
                    CompletableFuture<GetCommandInvocationResponse> invocationFuture
= getAsyncClient().getCommandInvocation(invocationRequest);
                    invocationFuture.whenComplete((commandInvocationResponse,
invocationEx) -> {
                        if (commandInvocationResponse != null) {
                            // 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);
                            }
                        } else {
                            Throwable invocationCause = (invocationEx instanceof
CompletionException) ? invocationEx.getCause() : invocationEx;
                            throw new CompletionException(invocationCause);
                        }
                    }).join();
                } catch (InterruptedException e) {
                    throw new RuntimeException(e);
                }
            } else {
```

```
                Throwable cause = (ex instanceof CompletionException) ?
ex.getCause() : ex;
                if (cause instanceof SsmException) {
                    throw (SsmException) cause;
                } else {
                    throw new RuntimeException(cause);
                }
            }
        }).join();

        return commandId[0];
    }

    /**
     * Creates an AWS SSM document asynchronously.
     *
     * @param docName The name of the document to create.
     * <p>
     * This method initiates an asynchronous request to create an SSM document.
     * If the request is successful, it prints the document status.
     * If an exception occurs, it handles the error appropriately.
     */
    public void createSSMDoc(String docName) throws SsmException {
        String jsonData = """
        {
        "schemaVersion": "2.2",
        "description": "Run a simple shell command",
        "mainSteps": [
            {
                "action": "aws:runShellScript",
                "name": "runEchoCommand",
                "inputs": {
                  "runCommand": [
                    "echo 'Hello, world!'"
                  ]
                }
              }
            ]
        }
        """;

        CreateDocumentRequest request = CreateDocumentRequest.builder()
                .content(jsonData)
                .name(docName)
```

```
                .documentType(DocumentType.COMMAND)
                .build();

        CompletableFuture<CreateDocumentResponse> future =
getAsyncClient().createDocument(request);
        future.thenAccept(response -> {
            System.out.println("The status of the SSM document is " +
response.documentDescription().status());
        }).exceptionally(ex -> {
            Throwable cause = (ex instanceof CompletionException) ? ex.getCause() :
ex;
            if (cause instanceof DocumentAlreadyExistsException) {
                throw new CompletionException(cause);
            } else if (cause instanceof SsmException) {
                throw new CompletionException(cause);
            } else {
                throw new RuntimeException(cause);
            }
        }).join();
    }

    /**
     * Updates an SSM maintenance window asynchronously.
     *
     * @param id The ID of the maintenance window to update.
     * @param name The new name for the maintenance window.
     * <p>
     * This method initiates an asynchronous request to update an SSM maintenance
window.
     * If the request is successful, it prints a success message.
     * If an exception occurs, it handles the error appropriately.
     */
    public void updateSSMMaintenanceWindow(String id, String name) throws
SsmException {
        UpdateMaintenanceWindowRequest updateRequest =
UpdateMaintenanceWindowRequest.builder()
                .windowId(id)
                .allowUnassociatedTargets(true)
                .duration(24)
                .enabled(true)
                .name(name)
                .schedule("cron(0 0 ? * MON *)")
                .build();
```

```
        CompletableFuture<UpdateMaintenanceWindowResponse> future =
getAsyncClient().updateMaintenanceWindow(updateRequest);
        future.whenComplete((response, ex) -> {
            if (response != null) {
                System.out.println("The SSM maintenance window was successfully
updated");
            } else {
                Throwable cause = (ex instanceof CompletionException) ?
ex.getCause() : ex;
                if (cause instanceof SsmException) {
                    throw new CompletionException(cause);
                } else {
                    throw new RuntimeException(cause);
                }
            }
        }).join();
    }

    /**
     * Creates an SSM maintenance window asynchronously.
     *
     * @param winName The name of the maintenance window.
     * @return The ID of the created or existing maintenance window.
     * <p>
     * This method initiates an asynchronous request to create an SSM maintenance
window.
     * If the request is successful, it prints the maintenance window ID.
     * If an exception occurs, it handles the error appropriately.
     */
    public String createMaintenanceWindow(String winName) throws SsmException,
DocumentAlreadyExistsException {
        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();

        CompletableFuture<CreateMaintenanceWindowResponse> future =
getAsyncClient().createMaintenanceWindow(request);
        final String[] windowId = {null};
```

```
        future.whenComplete((response, ex) -> {
            if (response != null) {
                String maintenanceWindowId = response.windowId();
                System.out.println("The maintenance window id is " +
maintenanceWindowId);
                windowId[0] = maintenanceWindowId;
            } else {
                Throwable cause = (ex instanceof CompletionException) ?
ex.getCause() : ex;
                if (cause instanceof DocumentAlreadyExistsException) {
                    throw new CompletionException(cause);
                } else if (cause instanceof SsmException) {
                    throw new CompletionException(cause);
                } else {
                    throw new RuntimeException(cause);
                }
            }
        }).join();

        if (windowId[0] == null) {
            MaintenanceWindowFilter filter = MaintenanceWindowFilter.builder()
                    .key("name")
                    .values(winName)
                    .build();

            DescribeMaintenanceWindowsRequest winRequest =
DescribeMaintenanceWindowsRequest.builder()
                    .filters(filter)
                    .build();

            CompletableFuture<DescribeMaintenanceWindowsResponse> describeFuture =
getAsyncClient().describeMaintenanceWindows(winRequest);
            describeFuture.whenComplete((describeResponse, describeEx) -> {
                if (describeResponse != null) {
                    List<MaintenanceWindowIdentity> windows =
describeResponse.windowIdentities();
                    if (!windows.isEmpty()) {
                        windowId[0] = windows.get(0).windowId();
                        System.out.println("Window ID: " + windowId[0]);
                    } else {
                        System.out.println("Window not found.");
                        windowId[0] = "";
                    }
                } else {
```

```
                    Throwable describeCause = (describeEx instanceof
 CompletionException) ? describeEx.getCause() : describeEx;
                    throw new RuntimeException("Error describing maintenance
 windows: " + describeCause.getMessage(), describeCause);
                }
            }).join();
        }

        return windowId[0];
    }
}
```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella Documentazione di riferimento API di AWS SDK for Java 2.x .
  - CreateDocument
  - CreateMaintenanceWindow
  - CreateOpsItem
  - DeleteMaintenanceWindow
  - ListCommandInvocations
  - SendCommand
  - UpdateOpsItem

Azioni

`CreateDocument`

Il seguente esempio di codice mostra come usare`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.

```
    /**
```

```
     * Creates an AWS SSM document asynchronously.
     *
     * @param docName The name of the document to create.
     * <p>
     * This method initiates an asynchronous request to create an SSM document.
     * If the request is successful, it prints the document status.
     * If an exception occurs, it handles the error appropriately.
     */
    public void createSSMDoc(String docName) throws SsmException {
        String jsonData = """
        {
        "schemaVersion": "2.2",
        "description": "Run a simple shell command",
        "mainSteps": [
            {
                "action": "aws:runShellScript",
                "name": "runEchoCommand",
                "inputs": {
                  "runCommand": [
                    "echo 'Hello, world!'"
                  ]
                }
              }
            ]
        }
        """;

        CreateDocumentRequest request = CreateDocumentRequest.builder()
                .content(jsonData)
                .name(docName)
                .documentType(DocumentType.COMMAND)
                .build();

        CompletableFuture<CreateDocumentResponse> future =
getAsyncClient().createDocument(request);
        future.thenAccept(response -> {
            System.out.println("The status of the SSM document is " +
response.documentDescription().status());
        }).exceptionally(ex -> {
            Throwable cause = (ex instanceof CompletionException) ? ex.getCause() :
ex;
            if (cause instanceof DocumentAlreadyExistsException) {
                throw new CompletionException(cause);
            } else if (cause instanceof SsmException) {
```

```
                throw new CompletionException(cause);
            } else {
                throw new RuntimeException(cause);
            }
        }).join();
    }
```

- Per i dettagli sull'API, consulta la CreateDocumentsezione 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.

```
    /**
     * Creates an SSM maintenance window asynchronously.
     *
     * @param winName The name of the maintenance window.
     * @return The ID of the created or existing maintenance window.
     * <p>
     * This method initiates an asynchronous request to create an SSM maintenance
window.
     * If the request is successful, it prints the maintenance window ID.
     * If an exception occurs, it handles the error appropriately.
     */
    public String createMaintenanceWindow(String winName) throws SsmException,
DocumentAlreadyExistsException {
        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();

        CompletableFuture<CreateMaintenanceWindowResponse> future =
getAsyncClient().createMaintenanceWindow(request);
        final String[] windowId = {null};
        future.whenComplete((response, ex) -> {
            if (response != null) {
                String maintenanceWindowId = response.windowId();
                System.out.println("The maintenance window id is " +
maintenanceWindowId);
                windowId[0] = maintenanceWindowId;
            } else {
                Throwable cause = (ex instanceof CompletionException) ?
ex.getCause() : ex;
                if (cause instanceof DocumentAlreadyExistsException) {
                    throw new CompletionException(cause);
                } else if (cause instanceof SsmException) {
                    throw new CompletionException(cause);
                } else {
                    throw new RuntimeException(cause);
                }
            }
        }).join();

        if (windowId[0] == null) {
            MaintenanceWindowFilter filter = MaintenanceWindowFilter.builder()
                    .key("name")
                    .values(winName)
                    .build();

            DescribeMaintenanceWindowsRequest winRequest =
DescribeMaintenanceWindowsRequest.builder()
                    .filters(filter)
                    .build();

            CompletableFuture<DescribeMaintenanceWindowsResponse> describeFuture =
getAsyncClient().describeMaintenanceWindows(winRequest);
            describeFuture.whenComplete((describeResponse, describeEx) -> {
                if (describeResponse != null) {
                    List<MaintenanceWindowIdentity> windows =
describeResponse.windowIdentities();
```

```
                    if (!windows.isEmpty()) {
                        windowId[0] = windows.get(0).windowId();
                        System.out.println("Window ID: " + windowId[0]);
                    } else {
                        System.out.println("Window not found.");
                        windowId[0] = "";
                    }
                } else {
                    Throwable describeCause = (describeEx instanceof
CompletionException) ? describeEx.getCause() : describeEx;
                    throw new RuntimeException("Error describing maintenance
windows: " + describeCause.getMessage(), describeCause);
                }
            }).join();
        }

        return windowId[0];
    }
```

- Per i dettagli sull'API, consulta la CreateMaintenanceWindowsezione 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.

```
    /**
     * Creates an SSM OpsItem asynchronously.
     *
     * @param title The title of the OpsItem.
     * @param source The source of the OpsItem.
     * @param category The category of the OpsItem.
```

```
     * @param severity The severity of the OpsItem.
     * @return The ID of the created OpsItem.
     * <p>
     * This method initiates an asynchronous request to create an SSM OpsItem.
     * If the request is successful, it returns the OpsItem ID.
     * If an exception occurs, it handles the error appropriately.
     */
    public String createSSMOpsItem(String title, String source, String category,
String severity) {
        CreateOpsItemRequest opsItemRequest = CreateOpsItemRequest.builder()
                .description("Created by the SSM Java API")
                .title(title)
                .source(source)
                .category(category)
                .severity(severity)
                .build();

        CompletableFuture<CreateOpsItemResponse> future =
getAsyncClient().createOpsItem(opsItemRequest);

        try {
            CreateOpsItemResponse response = future.join();
            return response.opsItemId();
        } catch (CompletionException e) {
            Throwable cause = e.getCause();
            if (cause instanceof SsmException) {
                throw (SsmException) cause;
            } else {
                throw new RuntimeException(cause);
            }
        }
    }
```

- Per i dettagli sull'API, consulta la CreateOpsItemsezione 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 SSM document asynchronously.
     *
     * @param documentName The name of the document to delete.
     * <p>
     * This method initiates an asynchronous request to delete an SSM document.
     * If an exception occurs, it handles the error appropriately.
     */
    public void deleteDoc(String documentName) {
        DeleteDocumentRequest documentRequest = DeleteDocumentRequest.builder()
                .name(documentName)
                .build();

        CompletableFuture<Void> future = CompletableFuture.runAsync(() -> {
            getAsyncClient().deleteDocument(documentRequest)
                    .thenAccept(response -> {
                        System.out.println("The SSM document was successfully
deleted.");
                    })
                    .exceptionally(ex -> {
                        throw new CompletionException(ex);
                    }).join();
        }).exceptionally(ex -> {
            Throwable cause = (ex instanceof CompletionException) ? ex.getCause() :
ex;
            if (cause instanceof SsmException) {
                throw new RuntimeException("SSM error: " + cause.getMessage(),
cause);
            } else {
                throw new RuntimeException("Unexpected error: " +
cause.getMessage(), cause);
            }
        });
```

```
        try {
            future.join();
        } catch (CompletionException ex) {
            throw ex.getCause() instanceof RuntimeException ? (RuntimeException)
ex.getCause() : ex;
        }
    }
```

- Per i dettagli sull'API, consulta la DeleteDocumentsezione 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.

```
    /**
     * Deletes an AWS SSM Maintenance Window asynchronously.
     *
     * @param winId The ID of the Maintenance Window to delete.
     * <p>
     * This method initiates an asynchronous request to delete an SSM Maintenance
Window.
     * If an exception occurs, it handles the error appropriately.
     */
    public void deleteMaintenanceWindow(String winId) {
        DeleteMaintenanceWindowRequest windowRequest =
DeleteMaintenanceWindowRequest.builder()
                .windowId(winId)
                .build();

        CompletableFuture<Void> future = CompletableFuture.runAsync(() -> {
            getAsyncClient().deleteMaintenanceWindow(windowRequest)
```

```
                    .thenAccept(response -> {
                        System.out.println("The maintenance window was successfully
deleted.");
                    })
                    .exceptionally(ex -> {
                        throw new CompletionException(ex);
                    }).join();
        }).exceptionally(ex -> {
            Throwable cause = (ex instanceof CompletionException) ? ex.getCause() :
ex;
            if (cause instanceof SsmException) {
                throw new RuntimeException("SSM error: " + cause.getMessage(),
cause);
            } else {
                throw new RuntimeException("Unexpected error: " +
cause.getMessage(), cause);
            }
        });

        try {
            future.join();
        } catch (CompletionException ex) {
            throw ex.getCause() instanceof RuntimeException ? (RuntimeException)
ex.getCause() : ex;
        }
    }
```

- Per i dettagli sull'API, consulta la DeleteMaintenanceWindowsezione 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.

```
    /**
     * Describes AWS SSM OpsItems asynchronously.
     *
     * @param key The key to filter OpsItems by (e.g., OPS_ITEM_ID).
     *
     * This method initiates an asynchronous request to describe SSM OpsItems.
     * If the request is successful, it prints the title and status of each OpsItem.
     * If an exception occurs, it handles the error appropriately.
     */
    public void describeOpsItems(String key) {
        OpsItemFilter filter = OpsItemFilter.builder()
                .key(OpsItemFilterKey.OPS_ITEM_ID)
                .values(key)
                .operator(OpsItemFilterOperator.EQUAL)
                .build();

        DescribeOpsItemsRequest itemsRequest = DescribeOpsItemsRequest.builder()
                .maxResults(10)
                .opsItemFilters(filter)
                .build();

        CompletableFuture<Void> future = CompletableFuture.runAsync(() -> {
            getAsyncClient().describeOpsItems(itemsRequest)
                    .thenAccept(itemsResponse -> {
                        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());
                        }
                    })
                    .exceptionally(ex -> {
                        throw new CompletionException(ex);
                    }).join();
        }).exceptionally(ex -> {
            Throwable cause = (ex instanceof CompletionException) ? ex.getCause() :
ex;
            if (cause instanceof SsmException) {
                throw new RuntimeException("SSM error: " + cause.getMessage(),
cause);
            } else {
                throw new RuntimeException("Unexpected error: " +
cause.getMessage(), cause);
```

```
            }
        });

        try {
            future.join();
        } catch (CompletionException ex) {
            throw ex.getCause() instanceof RuntimeException ? (RuntimeException)
ex.getCause() : ex;
        }
    }
```

- Per i dettagli sull'API, consulta la DescribeOpsItemssezione 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, consulta la DescribeParameterssezione 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, consulta la PutParametersezione 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 asynchronously.
    *
    * @param documentName The name of the document to use.
    * @param instanceId The ID of the instance to send the command to.
    * @return The command ID.
    * <p>
    * This method initiates asynchronous requests to send a SSM command to a
managed node.
    * It waits until the document is active, sends the command, and checks the
command execution status.
    */
   public String sendSSMCommand(String documentName, String instanceId) throws
InterruptedException, SsmException {
       // Before we use Document to send a command - make sure it is active.
       CompletableFuture<Void> documentActiveFuture = CompletableFuture.runAsync(()
-> {
           boolean isDocumentActive = false;
           DescribeDocumentRequest request = DescribeDocumentRequest.builder()
                   .name(documentName)
                   .build();

           while (!isDocumentActive) {
               CompletableFuture<DescribeDocumentResponse> response =
getAsyncClient().describeDocument(request);
               String documentStatus = response.join().document().statusAsString();
               if (documentStatus.equals("Active")) {
                   System.out.println("The SSM document is active and ready to
use.");
                   isDocumentActive = true;
               } else {
                   System.out.println("The SSM document is not active. Status: " +
documentStatus);
                   try {
```

```
                        Thread.sleep(5000);
                    } catch (InterruptedException e) {
                        throw new RuntimeException(e);
                    }
                }
            }
        });

        documentActiveFuture.join();

        // Create the SendCommandRequest.
        SendCommandRequest commandRequest = SendCommandRequest.builder()
                .documentName(documentName)
                .instanceIds(instanceId)
                .build();

        // Send the command.
        CompletableFuture<SendCommandResponse> commandFuture =
getAsyncClient().sendCommand(commandRequest);
        final String[] commandId = {null};

        commandFuture.whenComplete((commandResponse, ex) -> {
            if (commandResponse != null) {
                commandId[0] = commandResponse.command().commandId();
                System.out.println("Command ID: " + commandId[0]);

                // Wait for the command execution to complete.
                GetCommandInvocationRequest invocationRequest =
GetCommandInvocationRequest.builder()
                        .commandId(commandId[0])
                        .instanceId(instanceId)
                        .build();

                try {
                    System.out.println("Wait 5 secs");
                    TimeUnit.SECONDS.sleep(5);

                    // Retrieve the command execution details.
                    CompletableFuture<GetCommandInvocationResponse> invocationFuture
= getAsyncClient().getCommandInvocation(invocationRequest);
                    invocationFuture.whenComplete((commandInvocationResponse,
invocationEx) -> {
                        if (commandInvocationResponse != null) {
                            // 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);
                            }
                        } else {
                            Throwable invocationCause = (invocationEx instanceof
CompletionException) ? invocationEx.getCause() : invocationEx;
                            throw new CompletionException(invocationCause);
                        }
                    }).join();
                } catch (InterruptedException e) {
                    throw new RuntimeException(e);
                }
            } else {
                Throwable cause = (ex instanceof CompletionException) ?
ex.getCause() : ex;
                if (cause instanceof SsmException) {
                    throw (SsmException) cause;
                } else {
                    throw new RuntimeException(cause);
                }
            }
        }).join();

        return commandId[0];
    }
```

- Per i dettagli sull'API, consulta la SendCommandsezione 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.

```
   /**
    * Updates an SSM maintenance window asynchronously.
    *
    * @param id The ID of the maintenance window to update.
    * @param name The new name for the maintenance window.
    * <p>
    * This method initiates an asynchronous request to update an SSM maintenance
window.
    * If the request is successful, it prints a success message.
    * If an exception occurs, it handles the error appropriately.
    */
   public void updateSSMMaintenanceWindow(String id, String name) throws
SsmException {
       UpdateMaintenanceWindowRequest updateRequest =
UpdateMaintenanceWindowRequest.builder()
               .windowId(id)
               .allowUnassociatedTargets(true)
               .duration(24)
               .enabled(true)
               .name(name)
               .schedule("cron(0 0 ? * MON *)")
               .build();

       CompletableFuture<UpdateMaintenanceWindowResponse> future =
getAsyncClient().updateMaintenanceWindow(updateRequest);
       future.whenComplete((response, ex) -> {
           if (response != null) {
               System.out.println("The SSM maintenance window was successfully
updated");
           } else {
               Throwable cause = (ex instanceof CompletionException) ?
ex.getCause() : ex;
               if (cause instanceof SsmException) {
                   throw new CompletionException(cause);
```

```
                } else {
                    throw new RuntimeException(cause);
                }
            }
        }).join();
    }
```

- Per i dettagli sull'API, consulta la [UpdateMaintenanceWindow](sezione 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].

```
    /**
     * Resolves an AWS SSM OpsItem asynchronously.
     *
     * @param opsID The ID of the OpsItem to resolve.
     * <p>
     * This method initiates an asynchronous request to resolve an SSM OpsItem.
     * If an exception occurs, it handles the error appropriately.
     */
    public void resolveOpsItem(String opsID) {
        UpdateOpsItemRequest opsItemRequest = UpdateOpsItemRequest.builder()
                .opsItemId(opsID)
                .status(OpsItemStatus.RESOLVED)
                .build();

        CompletableFuture<Void> future = CompletableFuture.runAsync(() -> {
            getAsyncClient().updateOpsItem(opsItemRequest)
                    .thenAccept(response -> {
                        System.out.println("OpsItem resolved successfully.");
```

```
                    })
                    .exceptionally(ex -> {
                        throw new CompletionException(ex);
                    }).join();
        }).exceptionally(ex -> {
            Throwable cause = (ex instanceof CompletionException) ? ex.getCause() :
ex;
            if (cause instanceof SsmException) {
                throw new RuntimeException("SSM error: " + cause.getMessage(),
cause);
            } else {
                throw new RuntimeException("Unexpected error: " +
cause.getMessage(), cause);
            }
        });

        try {
            future.join();
        } catch (CompletionException ex) {
            throw ex.getCause() instanceof RuntimeException ? (RuntimeException)
ex.getCause() : ex;
        }
    }
```

- Per i dettagli sull'API, consulta la UpdateOpsItemsezione AWS SDK for Java 2.x API Reference.

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.

Gli scenari sono esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio o combinate con altri Servizi AWS.

Ogni esempio include un collegamento al codice sorgente completo, dove puoi trovare istruzioni su come configurare ed eseguire il codice nel contesto.

Argomenti

- Azioni
- Scenari

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, consulta la AnalyzeDocumentsezione 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 =
 textractClient.detectDocumentText(detectDocumentTextRequest);
            List<Block> docInfo = textResponse.blocks();
            for (Block block : docInfo) {
                System.out.println("The block type is " +
 block.blockType().toString());
            }

            DocumentMetadata documentMetadata = textResponse.documentMetadata();
            System.out.println("The number of pages in the document is " +
 documentMetadata.pages());

        } catch (TextractException | FileNotFoundException e) {

            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

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.textract.model.S3Object;
import software.amazon.awssdk.services.textract.TextractClient;
import software.amazon.awssdk.services.textract.model.Document;
import software.amazon.awssdk.services.textract.model.DetectDocumentTextRequest;
import software.amazon.awssdk.services.textract.model.DetectDocumentTextResponse;
import software.amazon.awssdk.services.textract.model.Block;
import software.amazon.awssdk.services.textract.model.DocumentMetadata;
import software.amazon.awssdk.services.textract.model.TextractException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DetectDocumentTextS3 {

    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <bucketName> <docName>\s

                Where:
                    bucketName - The name of the Amazon S3 bucket that contains the
 document.\s

                    docName - The document name (must be an image, i.e., book.png).
\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }
```

```
        String bucketName = args[0];
        String docName = args[1];
        Region region = Region.US_WEST_2;
        TextractClient textractClient = TextractClient.builder()
                .region(region)
                .build();

        detectDocTextS3(textractClient, bucketName, docName);
        textractClient.close();
    }

    public static void detectDocTextS3(TextractClient textractClient, String
bucketName, String docName) {
        try {
            S3Object s3Object = S3Object.builder()
                    .bucket(bucketName)
                    .name(docName)
                    .build();

            // Create a Document object and reference the s3Object instance.
            Document myDoc = Document.builder()
                    .s3Object(s3Object)
                    .build();

            DetectDocumentTextRequest detectDocumentTextRequest =
DetectDocumentTextRequest.builder()
                    .document(myDoc)
                    .build();

            DetectDocumentTextResponse textResponse =
textractClient.detectDocumentText(detectDocumentTextRequest);
            for (Block block : textResponse.blocks()) {
                System.out.println("The block type is " +
block.blockType().toString());
            }

            DocumentMetadata documentMetadata = textResponse.documentMetadata();
            System.out.println("The number of pages in the document is " +
documentMetadata.pages());

        } catch (TextractException e) {

            System.err.println(e.getMessage());
            System.exit(1);
```

```
        }
    }
}
```

- 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 (TextractException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        return "";
    }

    private static String getJobResults(TextractClient textractClient, String jobId)
{
        boolean finished = false;
        int index = 0;
        String status = "";

        try {
            while (!finished) {
                GetDocumentAnalysisRequest analysisRequest =
GetDocumentAnalysisRequest.builder()
                        .jobId(jobId)
                        .maxResults(1000)
                        .build();

                GetDocumentAnalysisResponse response =
textractClient.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, consulta la StartDocumentAnalysissezione AWS SDK for Java 2.x API Reference.

Scenari

Crea un'applicazione per analizzare il feedback dei clienti

L'esempio di codice seguente mostra come creare un'applicazione che analizza le schede dei commenti dei clienti, le traduce dalla loro lingua originale, ne determina il sentiment e genera un file audio dal testo tradotto.

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

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.

Gli scenari sono esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio o combinate con altri Servizi AWS.

Ogni esempio include un collegamento al codice sorgente completo, 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, consulta la ListTranscriptionJobssezione 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 di più 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);
                })
                .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 static Subscription currentSubscription;
        private final InputStream inputStream;

        private AudioStreamPublisher(InputStream inputStream) {
            this.inputStream = inputStream;
        }

        @Override
        public void subscribe(Subscriber<? super AudioStream> s) {

            if (currentSubscription == null) {
                currentSubscription = new SubscriptionImpl(s, inputStream);
            } else {
                currentSubscription.cancel();
                currentSubscription = new SubscriptionImpl(s, inputStream);
            }
            s.onSubscribe(currentSubscription);
        }
    }

    public static class SubscriptionImpl implements Subscription {
        private static final int CHUNK_SIZE_IN_BYTES = 1024;
        private final Subscriber<? super AudioStream> subscriber;
        private final InputStream inputStream;
        private final ExecutorService executor = Executors.newFixedThreadPool(1);
        private final 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 informazioni dettagliate sull'API, consulta i seguenti argomenti nella Documentazione di riferimento delle API AWS SDK for Java 2.x .
  - GetTranscriptionJob
  - StartTranscriptionJob

Esempi di Amazon Transcribe Streaming 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 Transcribe Streaming.

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.

Gli scenari sono esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio o combinate con altri Servizi AWS.

Ogni esempio include un collegamento al codice sorgente completo, dove puoi trovare istruzioni su come configurare ed eseguire il codice nel contesto.

Argomenti

- Azioni
- Scenari

Azioni

`StartMedicalStreamTranscription`

Il seguente esempio di codice mostra come utilizzare`StartMedicalStreamTranscription`.

SDK per Java 2.x

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
/*
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

This code demonstrates the process of starting a medical transcription job using the
 AWS Transcribe
Streaming service, including setting up the audio input stream, configuring the
 transcription request,
and handling the transcription response.
 */

public class TranscribeMedicalStreamingDemoApp {
    private static TranscribeStreamingAsyncClient client;

    public static void main(String args[])
        throws ExecutionException, InterruptedException, LineUnavailableException {

        client = TranscribeStreamingAsyncClient.builder()
            .credentialsProvider(getCredentials())
            .build();

        CompletableFuture<Void> result =
 client.startMedicalStreamTranscription(getMedicalRequest(16_000),
            new AudioStreamPublisher(getStreamFromMic()),
```

```
            getMedicalResponseHandler());

        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");
            throw new LineUnavailableException("The audio system microphone line is
not supported.");
        }

        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 StartMedicalStreamTranscriptionRequest getMedicalRequest(Integer
mediaSampleRateHertz) {
        return StartMedicalStreamTranscriptionRequest.builder()
            .languageCode(LanguageCode.EN_US.toString()) // For medical
transcription, EN_US is typically used.
            .mediaEncoding(MediaEncoding.PCM)
            .mediaSampleRateHertz(mediaSampleRateHertz)
            .specialty(Specialty.PRIMARYCARE) // Specify the medical specialty.
            .type(Type.CONVERSATION) // Set the type as CONVERSATION or DICTATION.
            .build();
    }
```

```
    private static StartMedicalStreamTranscriptionResponseHandler
 getMedicalResponseHandler() {
        return StartMedicalStreamTranscriptionResponseHandler.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 streamed successfully ===");
            })
            .subscriber(event -> {
                List<MedicalResult> results = ((MedicalTranscriptEvent)
 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();
        }
    }
}
```

- Per i dettagli sull'API, consulta la StartMedicalStreamTranscriptionsezione AWS SDK for Java 2.x API Reference.

`StartStreamTranscription`

Il seguente esempio di codice mostra come utilizzare`StartStreamTranscription`.

SDK per Java 2.x

> **Note**
>
> C'è 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);
                })
                .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 static Subscription currentSubscription;
        private final InputStream inputStream;

        private AudioStreamPublisher(InputStream inputStream) {
            this.inputStream = inputStream;
        }

        @Override
        public void subscribe(Subscriber<? super AudioStream> s) {

            if (currentSubscription == null) {
                currentSubscription = new SubscriptionImpl(s, inputStream);
            } else {
                currentSubscription.cancel();
                currentSubscription = new SubscriptionImpl(s, inputStream);
            }
            s.onSubscribe(currentSubscription);
        }
    }

    public static class SubscriptionImpl implements Subscription {
        private static final int CHUNK_SIZE_IN_BYTES = 1024;
        private final Subscriber<? super AudioStream> subscriber;
        private final InputStream inputStream;
        private final ExecutorService executor = Executors.newFixedThreadPool(1);
        private final 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, consulta la StartStreamTranscriptionsezione AWS SDK for Java 2.x API Reference.

Scenari

Trascrivi un file audio

Il seguente esempio di codice mostra come generare una trascrizione di un file audio sorgente utilizzando lo streaming di Amazon Transcribe.

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.

```
/**
 * 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();
        }
    }
}
```

- Per i dettagli sull'API, consulta la StartStreamTranscriptionsezione AWS SDK for Java 2.x API Reference.

Trascrivi l'audio da un microfono

Il seguente esempio di codice mostra come generare una trascrizione da un microfono utilizzando lo streaming di Amazon Transcribe.

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 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);
                })
                .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 static Subscription currentSubscription;
        private final InputStream inputStream;

        private AudioStreamPublisher(InputStream inputStream) {
            this.inputStream = inputStream;
        }

        @Override
        public void subscribe(Subscriber<? super AudioStream> s) {

            if (currentSubscription == null) {
                currentSubscription = new SubscriptionImpl(s, inputStream);
            } else {
                currentSubscription.cancel();
                currentSubscription = new SubscriptionImpl(s, inputStream);
            }
            s.onSubscribe(currentSubscription);
        }
    }

    public static class SubscriptionImpl implements Subscription {
        private static final int CHUNK_SIZE_IN_BYTES = 1024;
        private final Subscriber<? super AudioStream> subscriber;
        private final InputStream inputStream;
        private final ExecutorService executor = Executors.newFixedThreadPool(1);
        private final 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, consulta la StartStreamTranscriptionsezione AWS SDK for Java 2.x API Reference.

Esempi di Amazon Translate con SDK for Java 2.x

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando Amazon Translate. AWS SDK for Java 2.x

Gli scenari sono esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio o combinate con altri Servizi AWS.

Ogni esempio include un collegamento al codice sorgente completo, dove puoi trovare istruzioni su come configurare ed eseguire il codice nel contesto.

Argomenti

- Scenari

Scenari

Creazione di un chatbot Amazon Lex

Il seguente esempio di codice mostra come creare un chatbot 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 Amazon SNS

Il seguente esempio di codice mostra come creare un'applicazione con funzionalità di sottoscrizione e pubblicazione e tradurre 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, consultate 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 per analizzare il feedback dei clienti

L'esempio di codice seguente mostra come creare un'applicazione che analizza le schede dei commenti dei clienti, le traduce dalla loro lingua originale, ne determina il sentiment e genera un file audio dal testo tradotto.

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

Sicurezza per AWS SDK per 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 per 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 per Java 2.x

Il modello di responsabilità AWS condivisa modello di di si applica alla protezione dei dati in AWS SDK per 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 Per informazioni sull'utilizzo dei CloudTrail percorsi per acquisire AWS le attività, consulta Lavorare con i CloudTrail percorsi nella Guida per l'AWS CloudTrail utente.
- 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 140-3 per accedere AWS tramite un'interfaccia a riga di comando o un'API, usa un endpoint FIPS. Per ulteriori informazioni sugli endpoint FIPS disponibili, consulta il Federal Information Processing Standard (FIPS) 140-3.

Ti consigliamo 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 o. AWS SDKs 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 per 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 SSLContextdi per stampare quali versioni TLS/SSL sono supportate.

```
System.out.println(Arrays.toString(SSLContext.getDefault().getSupportedSSLParameters().getProto
```

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 i sistemi basati su OpenJDK JVMs, è possibile utilizzare la proprietà di sistema. `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 sul metodo consigliato per la firma delle richieste, consulta Signature Version 4 AWS per le richieste API nella Guida per l'utente IAM.

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)AWS in IAM nella Guida per l'utente 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. Conserva le credenziali dell'utente root e utilizzale 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 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 su IAM Identity Center, consulta Cos'è IAM Identity Center? 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, se si hanno 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 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, potresti avere un gruppo denominato IAMAdminse concedere a quel 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 Casi d'uso per utenti IAM nella Guida per l'utente 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. Per assumere temporaneamente un ruolo IAM in AWS Management Console, puoi passare da un ruolo utente a un ruolo IAM (console). 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 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 Create a role for a third-party identity provider (federation) nella Guida per l'utente 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 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 Accesso a risorse multi-account in IAM nella Guida per l'utente 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 normale che quel 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 inoltrato (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 an 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 Forward access sessions.
  - Ruolo di servizio: un ruolo di servizio è un ruolo IAM che un servizio assume per eseguire operazioni 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 Create a role to delegate permissions to an Servizio AWS nella Guida per l'utente 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'azione 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' EC2 istanza e che AWS CLI effettuano

richieste AWS API. Questa soluzione è preferibile alla memorizzazione delle chiavi di accesso all'interno dell' EC2 istanza. Per assegnare un AWS ruolo a un' EC2 istanza e renderlo disponibile per tutte le sue applicazioni, create un profilo di istanza collegato all'istanza. Un profilo di istanza contiene il ruolo e consente ai programmi in esecuzione sull' EC2 istanza di ottenere credenziali temporanee. Per ulteriori informazioni, consulta Utilizzare un ruolo IAM per concedere le autorizzazioni alle applicazioni in esecuzione su EC2 istanze Amazon nella IAM User Guide.

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 IAM.

Gli amministratori possono utilizzare le policy AWS JSON per specificare chi ha accesso a cosa. In altre parole, quale principale può eseguire operazioni 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 operazioni sulle risorse di cui hanno bisogno, un amministratore IAM può creare policy IAM. L'amministratore può quindi aggiungere le policy IAM ai ruoli e gli utenti possono assumere i ruoli.

Le policy IAM definiscono le autorizzazioni relative a un'operazione, a prescindere dal metodo utilizzato per eseguirla. Ad esempio, supponiamo di disporre di una policy che consente l'operazione `iam:GetRole`. Un utente con tale policy può ottenere informazioni sul ruolo dall' AWS Management Console AWS CLI, dall'o dall' AWS API.

Policy basate sull'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 ruolo IAM). Tali policy definiscono le operazioni che utenti e ruoli possono eseguire, su quali risorse e in quali condizioni. Per informazioni su come creare una policy basata su identità, consulta Definizione di autorizzazioni personalizzate IAM con policy gestite dal cliente nella Guida per l'utente 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 IAM.

Policy basate sulle risorse

Le policy basate su risorse sono documenti di policy JSON che è possibile collegare a una risorsa. Esempi di policy basate sulle 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 utilizzarli per controllare l'accesso a una risorsa specifica. Quando è collegata a una risorsa, una policy definisce le operazioni 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 () ACLs

Le liste di controllo degli accessi (ACLs) controllano quali principali (membri dell'account, utenti o ruoli) dispongono delle autorizzazioni per accedere a una risorsa. ACLs sono simili alle politiche basate sulle risorse, sebbene non utilizzino il formato del documento di policy JSON.

Amazon S3 e Amazon VPC sono esempi di servizi che supportano. AWS WAF ACLs Per ulteriori informazioni ACLs, consulta la panoramica della lista di controllo degli accessi (ACL) nella Amazon Simple Storage Service Developer Guide.

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 funzionalità 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 IAM.

- Politiche di controllo del servizio (SCPs): SCPs 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 di proprietà dell' Account AWS azienda. Se abiliti tutte le funzionalità di un'organizzazione, puoi applicare le politiche di controllo del servizio (SCPs) a uno o tutti i tuoi account. L'SCP limita le autorizzazioni per le entità presenti negli account dei membri, inclusa ciascuna di esse. Utente root dell'account AWS Per ulteriori informazioni su Organizations and SCPs, consulta le politiche di controllo dei servizi nella Guida AWS Organizations per l'utente.
- Politiche di controllo delle risorse (RCPs): RCPs sono politiche JSON che puoi utilizzare per impostare le autorizzazioni massime disponibili per le risorse nei tuoi account senza aggiornare le politiche IAM allegate a ciascuna risorsa di tua proprietà. L'RCP limita le autorizzazioni per le risorse negli account dei membri e può influire sulle autorizzazioni effettive per le identità, incluse le Utente root dell'account AWS, indipendentemente dal fatto che appartengano o meno all'organizzazione. Per ulteriori informazioni su Organizations e RCPs, incluso un elenco di Servizi AWS tale supporto RCPs, vedere Resource control policies (RCPs) nella Guida per l'AWS Organizations utente.
- 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 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 o meno una richiesta quando sono coinvolti più tipi di policy, consulta la 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 `awes:`*`GetWidget`* fittizie.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 awes: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 `awes:`*`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 i servizi che supportano politiche basate sulle risorse o liste di controllo degli accessi (ACLs), puoi utilizzare tali politiche 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 IAM.
- Per informazioni sulle differenze di utilizzo tra ruoli e policy basate su risorse per l'accesso multi-account, consulta Accesso a risorse multi-account in IAM 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:

- Governance e conformità per la sicurezza: queste guide all'implementazione di soluzioni illustrano considerazioni relative all'architettura e i passaggi per implementare le funzionalità di sicurezza e conformità.
- Riferimenti sui servizi conformi ai requisiti HIPAA: elenca i servizi HIPAA idonei. Non tutti Servizi AWS sono idonei alla normativa 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 per Java

La AWS SDK per 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
- Come migrare il codice da AWS SDK per Java 1.x a 2.x
- Cosa c'è di diverso tra AWS SDK per Java 1.x e 2.x
- Usa l'SDK per Java 1.x e 2.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 per Java 2.x
- Usa l'SDK
- Sicurezza per AWS SDK per Java

Come migrare il codice da AWS SDK per Java 1.x a 2.x

È possibile migrare le applicazioni SDK for Java 1.x esistenti in due modi.

1. Approccio automatizzato utilizzando lo strumento di migrazione.
2. Approccio manuale che sostituisce in modo incrementale le importazioni 1.x con le importazioni 2.x.

Ti consigliamo di iniziare utilizzando lo strumento di migrazione. Automatizza gran parte del lavoro sostitutivo di routine dal codice 1.x al 2.x.

Poiché la versione di anteprima dello strumento non migra tutte le funzionalità, dovrai cercare il codice v1 rimanente dopo aver eseguito lo strumento. Quando trovi codice che non è stato migrato dallo strumento, segui le step-by-step istruzioni (approccio manuale) e utilizza gli articoli della guida alla migrazione per completare la migrazione.

Argomenti

- Strumento di migrazione (versione di anteprima)
- step-by-stepIstruzioni di migrazione con esempio

Strumento di migrazione (versione di anteprima)

AWS SDK per Java Fornisce uno strumento di migrazione che aiuta ad automatizzare la migrazione del codice SDK for Java 1.x alla 2.x. Lo strumento utilizza OpenRewrite, uno strumento open source di refactoring del codice sorgente, per eseguire la migrazione.

Ora puoi usare lo strumento come versione di anteprima. Lo strumento supporta tutti i client di servizi SDK ad eccezione di Amazons3Client e di alto livello APIs come Dynamo. TransferManagerDBMapper Lo strumento presenta anche alcune limitazioni elencate alla fine di questo argomento.

Utilizza lo strumento di migrazione

Migra un progetto Maven

Segui le istruzioni riportate di seguito per migrare il tuo progetto SDK for Java 1.x basato su Maven utilizzando lo strumento plug-in Maven. OpenRewrite

1. Vai alla directory principale del tuo progetto Maven

   Apri una finestra di terminale (riga di comando) e vai alla directory principale dell'applicazione basata su Maven.

2. Esegui il comando del plugin `rewrite-maven-plugin`

   Puoi scegliere tra due modalità (obiettivi Maven): `dryRun` e. `run`

   Modalità **`dryRun`**

   In questa `dryRun` modalità, il plugin genera i diff log nell'output della console e un file di patch denominato `rewrite.patch` nella cartella. `target/rewrite` Questa modalità consente di visualizzare in anteprima le modifiche che verranno apportate, poiché non vengono apportate modifiche ai file del codice sorgente.

   L'esempio seguente mostra come richiamare il plugin in `dryRun` modalità.

```
mvn org.openrewrite.maven:rewrite-maven-plugin:<rewrite-plugin-version>*:dryRun \
  -Drewrite.recipeArtifactCoordinates=software.amazon.awssdk:v2-
migration:<sdkversion>**-PREVIEW \
  -Drewrite.activeRecipes=software.amazon.awssdk.v2migration.AwsSdkJavaV1ToV2
```

   *Sostituisci *`<rewrite-plugin-version>`* con il `rewriteMavenPluginVersion` valore che vedi in questo file di test.

   **Sostituisci *`<sdkversion>`* con una versione SDK 2.x. Visita Maven Central per verificare la versione più recente.

   > **Important**
   >
   > Assicurati di utilizzare la versione `rewrite-maven-plugin` mostrata nel file di test perché altre versioni potrebbero non funzionare.

L'output della console proveniente dalla `dryRun` modalità dovrebbe essere simile al seguente.

```
[WARNING] These recipes would make changes to project/src/test/resources/maven/
before/pom.xml:
[WARNING]     software.amazon.awssdk.v2migration.AwsSdkJavaV1ToV2
[WARNING]         software.amazon.awssdk.v2migration.UpgradeSdkDependencies
[WARNING]             org.openrewrite.java.dependencies.AddDependency:
 {groupId=software.amazon.awssdk, artifactId=apache-client, version=2.27.0,
 onlyIfUsing=com.amazonaws.ClientConfiguration}
[WARNING]             org.openrewrite.java.dependencies.AddDependency:
 {groupId=software.amazon.awssdk, artifactId=netty-nio-client, version=2.27.0,
 onlyIfUsing=com.amazonaws.ClientConfiguration}
[WARNING]             org.openrewrite.java.dependencies.ChangeDependency:
 {oldGroupId=com.amazonaws, oldArtifactId=aws-java-sdk-bom,
 newGroupId=software.amazon.awssdk, newArtifactId=bom, newVersion=2.27.0}
[WARNING]             org.openrewrite.java.dependencies.ChangeDependency:
 {oldGroupId=com.amazonaws, oldArtifactId=aws-java-sdk-s3,
 newGroupId=software.amazon.awssdk, newArtifactId=s3, newVersion=2.27.0}
[WARNING]             org.openrewrite.java.dependencies.ChangeDependency:
 {oldGroupId=com.amazonaws, oldArtifactId=aws-java-sdk-sqs,
 newGroupId=software.amazon.awssdk, newArtifactId=sqs, newVersion=2.27.0}
[WARNING] These recipes would make changes to project/src/test/resources/maven/
before/src/main/java/foo/bar/Application.java:
[WARNING]     software.amazon.awssdk.v2migration.AwsSdkJavaV1ToV2
[WARNING]         software.amazon.awssdk.v2migration.S3GetObjectConstructorToFluent
[WARNING]             software.amazon.awssdk.v2migration.ConstructorToFluent
[WARNING]         software.amazon.awssdk.v2migration.S3StreamingResponseToV2
[WARNING]         software.amazon.awssdk.v2migration.ChangeSdkType
[WARNING]         software.amazon.awssdk.v2migration.ChangeSdkCoreTypes
[WARNING]             software.amazon.awssdk.v2migration.ChangeExceptionTypes
[WARNING]                 org.openrewrite.java.ChangeType:
 {oldFullyQualifiedTypeName=com.amazonaws.AmazonClientException,
 newFullyQualifiedTypeName=software.amazon.awssdk.core.exception.SdkException}
[WARNING]                 org.openrewrite.java.ChangeMethodName:
 {methodPattern=com.amazonaws.AmazonServiceException getRequestId(),
 newMethodName=requestId}
[WARNING]                 org.openrewrite.java.ChangeMethodName:
 {methodPattern=com.amazonaws.AmazonServiceException getErrorCode(),
 newMethodName=awsErrorDetails().errorCode}
[WARNING]                 org.openrewrite.java.ChangeMethodName:
 {methodPattern=com.amazonaws.AmazonServiceException getServiceName(),
 newMethodName=awsErrorDetails().serviceName}
```

```
[WARNING]                org.openrewrite.java.ChangeMethodName:
 {methodPattern=com.amazonaws.AmazonServiceException getErrorMessage(),
 newMethodName=awsErrorDetails().errorMessage}
[WARNING]                org.openrewrite.java.ChangeMethodName:
 {methodPattern=com.amazonaws.AmazonServiceException getRawResponse(),
 newMethodName=awsErrorDetails().rawResponse().asByteArray}
[WARNING]                org.openrewrite.java.ChangeMethodName:
 {methodPattern=com.amazonaws.AmazonServiceException getRawResponseContent(),
 newMethodName=awsErrorDetails().rawResponse().asUtf8String}
[WARNING]                org.openrewrite.java.ChangeType:
 {oldFullyQualifiedTypeName=com.amazonaws.AmazonServiceException,
 newFullyQualifiedTypeName=software.amazon.awssdk.awscore.exception.AwsServiceException}
[WARNING]         software.amazon.awssdk.v2migration.NewClassToBuilderPattern
[WARNING]             software.amazon.awssdk.v2migration.NewClassToBuilder
[WARNING]             software.amazon.awssdk.v2migration.V1SetterToV2
[WARNING]         software.amazon.awssdk.v2migration.V1GetterToV2
...
[WARNING]         software.amazon.awssdk.v2migration.V1BuilderVariationsToV2Builder
[WARNING]         software.amazon.awssdk.v2migration.NewClassToBuilderPattern
[WARNING]             software.amazon.awssdk.v2migration.NewClassToBuilder
[WARNING]             software.amazon.awssdk.v2migration.V1SetterToV2
[WARNING]         software.amazon.awssdk.v2migration.HttpSettingsToHttpClient
[WARNING]         software.amazon.awssdk.v2migration.WrapSdkClientBuilderRegionStr
[WARNING] Patch file available:
[WARNING]     project/src/test/resources/maven/before/target/rewrite/rewrite.patch
[WARNING] Estimate time saved: 20m
[WARNING] Run 'mvn rewrite:run' to apply the recipes.
```

Modalità `run`

Quando esegui il plugin in `run` modalità, modifica il codice sorgente sul disco per applicare le modifiche. Assicurati di avere un backup del codice sorgente prima di eseguire il comando.

L'esempio seguente mostra come richiamare il plugin in `run` modalità.

```
mvn org.openrewrite.maven:rewrite-maven-plugin:<rewrite-plugin-version>*:run \
  -Drewrite.recipeArtifactCoordinates=software.amazon.awssdk:v2-
migration:<sdkversion>**-PREVIEW \
  -Drewrite.activeRecipes=software.amazon.awssdk.v2migration.AwsSdkJavaV1ToV2
```

*`<rewrite-plugin-version>`*Sostituiscilo con `rewriteMavenPluginVersion`value quello che vedi in questo file di test.

**Sostituisci *<sdkversion>* con una versione SDK 2.x. Visita Maven Central per verificare la versione più recente.

Dopo aver eseguito il comando, compila l'applicazione ed esegui dei test per verificare le modifiche.

Migra un progetto Gradle

Segui le istruzioni seguenti per migrare il tuo progetto SDK for Java 1.x basato su Gradle utilizzando lo strumento plug-in Gradle. OpenRewrite

1. Vai alla directory principale del tuo progetto Gradle

   Apri una finestra di terminale (riga di comando) e vai alla directory principale dell'applicazione basata su Gradle.

2. Crea uno script di init Gradle

   Crea un `init.gradle` file con il seguente contenuto nella directory.

```
initscript {
    repositories {
        maven { url "https://plugins.gradle.org/m2" }
    }
    dependencies {
        classpath("org.openrewrite:plugin:<rewrite-plugin-version>*")
    }
}

rootProject {
    plugins.apply(org.openrewrite.gradle.RewritePlugin)
    dependencies {
        rewrite("software.amazon.awssdk:v2-migration:latest.release")
    }

    afterEvaluate {
        if (repositories.isEmpty()) {
            repositories {
                mavenCentral()
            }
        }
    }
```

```
}
```

*Sostituiscilo *<rewrite-plugin-version>* con la versione che vedi in questo file di test.

3. Esegui il `rewrite` comando

   Come con il plug-in Maven, puoi eseguire il plug-in Gradle in `dryRun` modalità o. `run`

   Modalità **`dryRun`**

   L'esempio seguente mostra come richiamare il plugin in modalità. `dryRun`

```
gradle rewriteDryRun --init-script init.gradle \
  -Drewrite.activeRecipes=software.amazon.awssdk.v2migration.AwsSdkJavaV1ToV2
```

   Modalità **`run`**

   L'esempio seguente mostra come richiamare il plugin in `run` modalità.

```
gradle rewriteRun --init-script init.gradle \
  -Drewrite.activeRecipes=software.amazon.awssdk.v2migration.AwsSdkJavaV1ToV2
```

Limitazioni attuali

L'attuale versione di anteprima non supporta la migrazione di tutte le funzionalità dell'SDK V1. Stiamo aggiungendo il supporto per le funzionalità in modo incrementale.

Se lo strumento non è in grado di migrare un metodo o una classe da V1 a V2, l'output V2 contiene un commento che inizia con:

```
/*AWS SDK for Java v2 migration: Transform for ...
```

Dopo il commento, lo strumento emette uno stub generico della versione V2 del metodo o della classe. Ad esempio, nel seguente output, lo strumento di migrazione ha tentato di migrare il metodo del client V1 S3: `setBucketLifecycleConfiguration`

```
/*AWS SDK for Java v2 migration: Transform for setBucketLifecycleConfiguration method
 not supported.
Please manually migrate your code by using builder pattern, update from
 BucketLifecycleConfiguration.Rule
```

```
to LifecycleRule, StorageClass to TransitionStorageClass, and adjust imports and
 names.*/
s3.putBucketLifecycleConfiguration(
        PutBucketLifecycleConfigurationRequest.builder()
            .bucket(bucketName)
            .lifecycleConfiguration(BucketLifecycleConfiguration.builder()
                .build())
            .build());
```

I collegamenti nell'elenco seguente rimandano alle informazioni sulla migrazione per aiutarti a migrare manualmente il codice.

- the section called “Differenze tra i client S3”
- S3 Transfer Manager () TransferManager
- Mappatura degli oggetti DynamoDB (Dynamo) DBMapper
- EC2 utilità per i metadati EC2 MetadataUtils ()
- Camerieri () AmazonDynamo DBWaiters
- IAM Policy Builder (politica)
- CloudFront preassegnazione (,) CloudFrontUrlSigner CloudFrontCookieSigner
- Notifiche di eventi S3 (S3) EventNotification
- Pubblicazione delle metriche SDK (documentazione 1.x, documentazione 2.x)

step-by-stepIstruzioni 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 un livello superiore APIs 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>2.27.21</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_ IDs utilizzato nelle importazioni v1, troverete l'unico SERVICE_ utilizzato. IDs 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 tutte le versioni univoche di V1 SERVICE_IDs, è possibile determinare l'artefatto Maven corrispondente per la dipendenza v2 facendo riferimento a. the section called “Dal nome del pacchetto alle mappature ArtifactID”

4. Aggiungi 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>2.27.21</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 `cutsort`, e `uniq` i comandi per isolare SERVICE_. IDs

```
rg --no-filename 'import\s+com\.amazonaws\.services' | cut -d '.' -f 4 | sort | uniq
```

Per questa applicazione, i seguenti SERVICE_ IDs sono 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. Dobbiamo solo trovare i SERVICE_ IDs che vengono 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

Il SERVICE_ IDs per la v1 che abbiamo isolato dalla Fase 2, ad esempio `autoscaling` e, `cloudformation` può essere mappato 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. ec2.*`
	ec2
	
	`software.amazon.awssdk.services. ec2.*`
		`<dependency>`
		`  <groupId>software.amazon.awssdk</groupId>`
		`  <artifactId> ec2</artifactId>`
		`</dependency>`

scalabilità automatica `com.amazonaws.services. autoscaling .*`	scalabilità automatica `software.amazon.awssdk.services. autoscaling .*`	`<dependency>` `  <groupId>software.amazon.awssdk</groupId>`

v1 SERVICE_ID è mappato a... nome del pacchetto	v2 SERVICE_ID mappa su... nome del pacchetto	dipendenza da Maven v2
		`<artifactId> autoscaling </artifactId>` `</dependency>`

cloudformation
 	
`com.amazonaws.services. cloudformation .*`
	cloudformation
	
	`software.amazon.awssdk. cloudformation .*`
		`<dependency>`
		`<groupId>software.amazon.awssdk</groupId>`
		`<artifactId> cloudformation </artifactId>`
		`</dependency>`

gestione dell'identità*
 	
`com.amazonaws.services. identitymanagement .*`
	sono*
	
	`software.amazon.awssdk. iam.*`
		`<dependency>`
		`<groupId>software.amazon.awssdk</groupId>`
		`<artifactId> iam</artifactId>`
		`</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 cliente EC2 di servizio di`com.amazonaws.services.ec2.AmazonEC2Client`.
- Diverse classi di EC2 modelli utilizzate. 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.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 aEc2Client`.
- 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 v2`Ec2Client`, 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
`import com.amazonaws.services.ec2.model.DescribeInstancesRequest`	`import software.amazon.awssdk.services.ec2.model.DescribeInstancesRequest`

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
`final DescribeInstancesRequest request = new DescribeInstancesRequest()`	`final DescribeInstancesRequest request = DescribeInstancesRequest.builder()` `.instanceIds(instanceIdsCopy)`

modifica	in
`.withInstanceIds(instanceIdsCopy);`	`.build();`

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'`request`istanza, 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 eseguivamo 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 impaginatore 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 “Step-by-step istruzioni” (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 SDKs utilizzata 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)
com.amazonaws.services.service_id	aws-java-sdk-ID_SERVIZIO	ID_SERVIZIO	software.amazon.awssdk.services.service_id

nome del pacchetto v1 (dichiarazione di importazione)	artifactID v1	artifactID v2	nome del pacchetto v2 (istruzione di importazione)
Esempio di Amazon Cognito Identity (SERVICE_ID:) ***`cognitoidentity`***			
com.amazonaws.services. identità cognitiva	aws-java-sdk- identità cognitiva	identità cognitiva	software.amazon.awssdk.services. identità cognitiva

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_ IDs differiscono tra quelli mostrati nella SDKs 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 `com.amazonaws.serv`	Tutti gli ArtifactID sono racchiusi in tag come	Tutti gli ArtifactID sono racchiusi tra tag come	Tutti i nomi dei pacchetti iniziano con `software.amazon.aw`

Nome servizio	nome del pacchetto v1	artifactID v1	artifactID v2	nome del pacchetto v2
	`ices` quello mostrato nella prima riga.	mostrato nella prima riga.	mostrato nella prima riga.	`ssdk` quello mostrato nella prima riga.
API Gateway	com.amazonaws.services.apigateway	<artifactId>aws-java-sdk-gateway API</artifactId>	<artifactId>una porta d'accesso </artifactId>	software.amazon.awssdk.services.apigateway
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	scoperta delle applicazioni
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

Nome servizio	nome del pacchetto v1	artifactID v1	artifactID v2	nome del pacchetto v2
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
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	elasticma preduce	emr	emr	emr
Glue DataBrew	incolla databrew	incollare databrew	databrew	databrew
IAM Roles Anywhere	I miei ruoli sono ovunque	sono ruoli ovunque	rolesanywhere	rolesanywhere

Nome servizio	nome del pacchetto v1	artifactID v1	artifactID v2	nome del pacchetto v2
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
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 gestita	grafana	grafana
Mechanical Turk	mturk	richiedente turco meccanico	mturk	mturk
Suggerimenti di strategia dell'Hub di migrazione	raccomandazioni sulla strategia dell'hub di migrazione	raccomandazioni strategiche di migrationhub	strategia migrationhub	strategia migrationhub

Nome servizio	nome del pacchetto v1	artifactID v1	artifactID v2	nome del pacchetto v2
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
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 per 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 per 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 per 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>2.27.21</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.

Immutabile POJOs

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

Metodi Setter e Getter

Nella versione AWS SDK per 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
Dinamo DBMapper	DynamoDbEnhancedClient	2.12.0
Waiter	Waiter	2.15.0
CloudFrontUrlSigner, CloudFrontCookieSigner	CloudFrontUtilities	2.18.33
TransferManager	S3TransferManager	2.19.0
EC2 Client per metadati	EC2 Client per metadati	2.19.29
Analizzatore URI S3	Analizzatore URI S3	2.20.41
IAM Policy Builder	IAM Policy Builder	2.20.126
Notifiche di eventi di Amazon S3	Notifiche di eventi S3	2.25.11
Buffering lato client di Amazon SQS	API di batch automatico delle richieste per Amazon SQS	2.28.0
Listener di avanzamento	Listener di avanzamento	non ancora rilasciato

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
Client di crittografia Amazon S3	3.0.0 1
AWS Client di crittografia del database per DynamoDB	3.0.0 2

[1] 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>
```

[2] 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

- Transfer Manager
- EC2 utilità per i metadati
- CloudFrontpreassegnazione
- analisi URI S3
- Mappatura/documento DynamoDB APIs

- IAM Policy Builder
- Notifiche di eventi S3
- Pubblicazione delle metriche SDK (documentazione 1.x, documentazione 2.x)

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
AWSACMPCAAsyncClient
```

Example dei nomi delle classi in 2.x

```
DynamoDbClient
AcmAsyncClient
```

Modifiche al nome della classe client

Client 1.x	Client 2.x
com.amazonaws.services.acmpca.AWSACMPCAAsyncClient	software.amazon.awssdk.services.acm.AcmAsyncClient
com.amazonaws.services.acmpca.AWSACMPCAClient	software.amazon.awssdk.services.acm.AcmClient
com.amazonaws.services.alexaforbusiness.AmazonAlexaForBusinessAsyncClient	software.amazon.awssdk.services.alexaforbusiness.AlexaForBusinessAsyncClient
com.amazonaws.services.alexaforbusiness.AmazonAlexaForBusinessClient	software.amazon.awssdk.services.alexaforbusiness.AlexaForBusinessClient
com.amazonaws.services.apigateway.AmazonApiGatewayAsyncClient	software.amazon.awssdk.services.apigateway.ApiGatewayAsyncClient
com.amazonaws.services.apigateway.AmazonApiGatewayClient	software.amazon.awssdk.services.apigateway.ApiGatewayClient
com.amazonaws.services.applicationautoscaling.AWSApplicationAutoScalingAsyncClient	software.amazon.awssdk.services.applicationautoscaling.ApplicationAutoScalingAsyncClient
com.amazonaws.services.applicationautoscaling.AWSApplicationAutoScalingClient	software.amazon.awssdk.services.applicationautoscaling.ApplicationAutoScalingClient
com.amazonaws.services.applicationdiscovery.AWSApplicationDiscoveryAsyncClient	software.amazon.awssdk.services.applicationdiscovery.ApplicationDiscoveryAsyncClient

Client 1.x	Client 2.x
com.amazonaws.services.applicationdiscovery.AWSApplicationDiscoveryClient	software.amazon.awssdk.services.applicationdiscovery.ApplicationDiscoveryClient
com.amazonaws.services.appstream.AmazonAppStreamAsyncClient	software.amazon.awssdk.services.appstream.AppStreamAsyncClient
com.amazonaws.services.appstream.AmazonAppStreamClient	software.amazon.awssdk.services.appstream.AppStreamClient
com.amazonaws.services.appsync.AWSAppSyncAsyncClient	software.amazon.awssdk.services.appsync.AppSyncAsyncClient
com.amazonaws.services.appsync.AWSAppSyncClient	software.amazon.awssdk.services.appsync.AppSyncClient
com.amazonaws.services.athena.AmazonAthenaAsyncClient	software.amazon.awssdk.services.athena.AthenaAsyncClient
com.amazonaws.services.athena.AmazonAthenaClient	software.amazon.awssdk.services.athena.AthenaClient
com.amazonaws.services.autoscaling.AmazonAutoScalingAsyncClient	software.amazon.awssdk.services.autoscaling.AutoScalingAsyncClient
com.amazonaws.services.autoscaling.AmazonAutoScalingClient	software.amazon.awssdk.services.autoscaling.AutoScalingClient
com.amazonaws.services.autoscalingplans.AWSAutoScalingPlansAsyncClient	software.amazon.awssdk.services.autoscalingplans.AutoScalingPlansAsyncClient

Client 1.x	Client 2.x
com.amazonaws.services.autoscalingplans.AWSAutoScalingPlansClient	software.amazon.awssdk.services.autoscalingplans.AutoScalingPlansClient
com.amazonaws.services.batch.AWSBatchAsyncClient	software.amazon.awssdk.services.batch.BatchAsyncClient
com.amazonaws.services.batch.AWSBatchClient	software.amazon.awssdk.services.batch.BatchClient
com.amazonaws.services.budgets.AWSBudgetsAsyncClient	software.amazon.awssdk.services.budgets.BudgetsAsyncClient
com.amazonaws.services.budgets.AWSBudgetsClient	software.amazon.awssdk.services.budgets.BudgetsClient
com.amazonaws.services.certificatemanager.AWSCertificateManagerAsyncClient	software.amazon.awssdk.services.acm.AcmAsyncClient
com.amazonaws.services.certificatemanager.AWSCertificateManagerClient	software.amazon.awssdk.services.acm.AcmClient
com.amazonaws.services.cloud9.AWSCloud9AsyncClient	software.amazon.awssdk.services.cloud9.Cloud9AsyncClient
com.amazonaws.services.cloud9.AWSCloud9Client	software.amazon.awssdk.services.cloud9.Cloud9Client
com.amazonaws.services.clouddirectory.AmazonCloudDirectoryAsyncClient	software.amazon.awssdk.services.clouddirectory.CloudDirectoryAsyncClient
com.amazonaws.services.clouddirectory.AmazonCloudDirectoryClient	software.amazon.awssdk.services.clouddirectory.CloudDirectoryClient

Client 1.x	Client 2.x
com.amazonaws.services.cloudformation.AmazonCloudFormationAsyncClient	software.amazon.awssdk.services.cloudformation.CloudFormationAsyncClient
com.amazonaws.services.cloudformation.AmazonCloudFormationClient	software.amazon.awssdk.services.cloudformation.CloudFormationClient
com.amazonaws.services.cloudfront.AmazonCloudFrontAsyncClient	software.amazon.awssdk.services.cloudfront.CloudFrontAsyncClient
com.amazonaws.services.cloudfront.AmazonCloudFrontClient	software.amazon.awssdk.services.cloudfront.CloudFrontClient
com.amazonaws.services.cloudhsm.AWSCloudHSMAsyncClient	software.amazon.awssdk.services.cloudhsm.CloudHsmAsyncClient
com.amazonaws.services.cloudhsm.AWSCloudHSMClient	software.amazon.awssdk.services.cloudhsm.CloudHsmClient
com.amazonaws.services.cloudhsmv2.AWSCloudHSMV2AsyncClient	software.amazon.awssdk.services.cloudhsmv2.CloudHsmV2AsyncClient
com.amazonaws.services.cloudhsmv2.AWSCloudHSMV2Client	software.amazon.awssdk.services.cloudhsmv2.CloudHsmV2Client
com.amazonaws.services.cloudsearchdomain.AmazonCloudSearchDomainAsyncClient	software.amazon.awssdk.services.cloudsearchdomain.CloudSearchDomainAsyncClient

Client 1.x	Client 2.x
com.amazonaws.services.cloudsearchdomain.AmazonCloudSearchDomainClient	software.amazon.awssdk.services.cloudsearchdomain.CloudSearchDomainClient
com.amazonaws.services.cloudsearchv2.AmazonCloudSearchAsyncClient	software.amazon.awssdk.services.cloudsearch.CloudSearchAsyncClient
com.amazonaws.services.cloudsearchv2.AmazonCloudSearchClient	software.amazon.awssdk.services.cloudsearch.CloudSearchClient
com.amazonaws.services.cloudtrail.AWSCloudTrailAsyncClient	software.amazon.awssdk.services.cloudtrail.CloudTrailAsyncClient
com.amazonaws.services.cloudtrail.AWSCloudTrailClient	software.amazon.awssdk.services.cloudtrail.CloudTrailClient
com.amazonaws.services.cloudwatch.AmazonCloudWatchAsyncClient	software.amazon.awssdk.services.cloudwatch.CloudWatchAsyncClient
com.amazonaws.services.cloudwatch.AmazonCloudWatchClient	software.amazon.awssdk.services.cloudwatch.CloudWatchClient
com.amazonaws.services.cloudwatchevents.AmazonCloudWatchEventsAsyncClient	software.amazon.awssdk.services.cloudwatchevents.CloudWatchEventsAsyncClient
com.amazonaws.services.cloudwatchevents.AmazonCloudWatchEventsClient	software.amazon.awssdk.services.cloudwatchevents.CloudWatchEventsClient

Client 1.x	Client 2.x
com.amazonaws.services.codebuild.AWSCodeBuildAsyncClient	software.amazon.awssdk.services.codebuild.CodeBuildAsyncClient
com.amazonaws.services.codebuild.AWSCodeBuildClient	software.amazon.awssdk.services.codebuild.CodeBuildClient
com.amazonaws.services.codecommit.AWSCodeCommitAsyncClient	software.amazon.awssdk.services.codecommit.CodeCommitAsyncClient
com.amazonaws.services.codecommit.AWSCodeCommitClient	software.amazon.awssdk.services.codecommit.CodeCommitClient
com.amazonaws.services.codedeploy.AmazonCodeDeployAsyncClient	software.amazon.awssdk.services.codedeploy.CodeDeployAsyncClient
com.amazonaws.services.codedeploy.AmazonCodeDeployClient	software.amazon.awssdk.services.codedeploy.CodeDeployClient
com.amazonaws.services.codepipeline.AWSCodePipelineAsyncClient	software.amazon.awssdk.services.codepipeline.CodePipelineAsyncClient
com.amazonaws.services.codepipeline.AWSCodePipelineClient	software.amazon.awssdk.services.codepipeline.CodePipelineClient
com.amazonaws.services.codestar.AWSCodeStarAsyncClient	software.amazon.awssdk.services.codestar.CodeStarAsyncClient
com.amazonaws.services.codestar.AWSCodeStarClient	software.amazon.awssdk.services.codestar.CodeStarClient

Client 1.x	Client 2.x
com.amazonaws.services.cognitoidentity.AmazonCognitoIdentityAsyncClient	software.amazon.awssdk.services.cognitoidentity.CognitoIdentityAsyncClient
com.amazonaws.services.cognitoidentity.AmazonCognitoIdentityClient	software.amazon.awssdk.services.cognitoidentity.CognitoIdentityClient
com.amazonaws.services.cognitoidp.AWSCognitoIdentityProviderAsyncClient	software.amazon.awssdk.services.cognitoidentityprovider.CognitoIdentityProviderAsyncClient
com.amazonaws.services.cognitoidp.AWSCognitoIdentityProviderClient	software.amazon.awssdk.services.cognitoidentityprovider.CognitoIdentityProviderClient
com.amazonaws.services.cognitosync.AmazonCognitoSyncAsyncClient	software.amazon.awssdk.services.cognitosync.CognitoSyncAsyncClient
com.amazonaws.services.cognitosync.AmazonCognitoSyncClient	software.amazon.awssdk.services.cognitosync.CognitoSyncClient
com.amazonaws.services.comprehend.AmazonComprehendAsyncClient	software.amazon.awssdk.services.comprehend.ComprehendAsyncClient
com.amazonaws.services.comprehend.AmazonComprehendClient	software.amazon.awssdk.services.comprehend.ComprehendClient
com.amazonaws.services.config.AmazonConfigAsyncClient	software.amazon.awssdk.services.config.ConfigAsyncClient

Client 1.x	Client 2.x
com.amazonaws.services.config.AmazonConfigClient	software.amazon.awssdk.services.config.ConfigClient
com.amazonaws.services.connect.AmazonConnectAsyncClient	software.amazon.awssdk.services.connect.ConnectAsyncClient
com.amazonaws.services.connect.AmazonConnectClient	software.amazon.awssdk.services.connect.ConnectClient
com.amazonaws.services.costandusagereport.AWSCostAndUsageReportAsyncClient	software.amazon.awssdk.services.costandusagereport.CostAndUsageReportAsyncClient
com.amazonaws.services.costandusagereport.AWSCostAndUsageReportClient	software.amazon.awssdk.services.costandusagereport.CostAndUsageReportClient
com.amazonaws.services.costexplorer.AWSCostExplorerAsyncClient	software.amazon.awssdk.services.costexplorer.CostExplorerAsyncClient
com.amazonaws.services.costexplorer.AWSCostExplorerClient	software.amazon.awssdk.services.costexplorer.CostExplorerClient
com.amazonaws.services.databasemigrationservice.AWSDatabaseMigrationServiceAsyncClient	software.amazon.awssdk.services.databasemigration.DatabaseMigrationAsyncClient
com.amazonaws.services.databasemigrationservice.AWSDatabaseMigrationServiceClient	software.amazon.awssdk.services.databasemigration.DatabaseMigrationClient
com.amazonaws.services.datapipeline.DataPipelineAsyncClient	software.amazon.awssdk.services.datapipeline.DataPipelineAsyncClient

Client 1.x	Client 2.x
com.amazonaws.services.datapipeline.DataPipelineClient	software.amazon.awssdk.services.datapipeline.DataPipelineAsyncClient
com.amazonaws.services.dax.AmazonDaxAsyncClient	software.amazon.awssdk.services.dax.DaxAsyncClient
com.amazonaws.services.dax.AmazonDaxClient	software.amazon.awssdk.services.dax.DaxClient
com.amazonaws.services.devicefarm.AWSDeviceFarmAsyncClient	software.amazon.awssdk.services.devicefarm.DeviceFarmAsyncClient
com.amazonaws.services.devicefarm.AWSDeviceFarmClient	software.amazon.awssdk.services.devicefarm.DeviceFarmClient
com.amazonaws.services.directconnect.AmazonDirectConnectAsyncClient	software.amazon.awssdk.services.directconnect.DirectConnectAsyncClient
com.amazonaws.services.directconnect.AmazonDirectConnectClient	software.amazon.awssdk.services.directconnect.DirectConnectClient
com.amazonaws.services.directory.AWSDirectoryServiceAsyncClient	software.amazon.awssdk.services.directory.DirectoryAsyncClient
com.amazonaws.services.directory.AWSDirectoryServiceClient	software.amazon.awssdk.services.directory.DirectoryClient
com.amazonaws.services.dlm.AmazonDLMAsyncClient	software.amazon.awssdk.services.dlm.DlmAsyncClient

Client 1.x	Client 2.x
com.amazonaws.services.dlm.AmazonDLMClient	software.amazon.awssdk.services.dlm.DlmClient
com.amazonaws.services.dynamodbv2.AmazonDynamoDBAsyncClient	software.amazon.awssdk.services.dynamodb.DynamoDbAsyncClient
com.amazonaws.services.dynamodbv2.AmazonDynamoDBClient	software.amazon.awssdk.services.dynamodb.DynamoDbClient
com.amazonaws.services.dynamodbv2.AmazonDynamoDBStreamsAsyncClient	software.amazon.awssdk.services.dynamodb.streams.DynamoDbStreamsAsyncClient
com.amazonaws.services.dynamodbv2.AmazonDynamoDBStreamsClient	software.amazon.awssdk.services.dynamodb.streams.DynamoDbStreamsClient
com.amazonaws.services.ec2.AmazonEC2AsyncClient	software.amazon.awssdk.services.ec2.Ec2AsyncClient
com.amazonaws.services.ec2.AmazonEC2Client	software.amazon.awssdk.services.ec2.Ec2Client
com.amazonaws.services.ecr.AmazonECRAsyncClient	software.amazon.awssdk.services.ecr.EcrAsyncClient
com.amazonaws.services.ecr.AmazonECRClient	software.amazon.awssdk.services.ecr.EcrClient
com.amazonaws.services.ecs.AmazonECSAsyncClient	software.amazon.awssdk.services.ecs.EcsAsyncClient
com.amazonaws.services.ecs.AmazonECSClient	software.amazon.awssdk.services.ecs.EcsClient

Client 1.x	Client 2.x
com.amazonaws.services.eks.AmazonEKSAsyncClient	software.amazon.awssdk.services.eks.EksAsyncClient
com.amazonaws.services.eks.AmazonEKSClient	software.amazon.awssdk.services.eks.EksClient
com.amazonaws.services.elasticache.AmazonElastiCacheAsyncClient	software.amazon.awssdk.services.elasticache.ElastiCacheAsyncClient
com.amazonaws.services.elasticache.AmazonElastiCacheClient	software.amazon.awssdk.services.elasticache.ElastiCacheClient
com.amazonaws.services.elasticbeanstalk.AWSElasticBeanstalkAsyncClient	software.amazon.awssdk.services.elasticbeanstalk.ElasticBeanstalkAsyncClient
com.amazonaws.services.elasticbeanstalk.AWSElasticBeanstalkClient	software.amazon.awssdk.services.elasticbeanstalk.ElasticBeanstalkClient
com.amazonaws.services.elasticfilesystem.AmazonElasticFileSystemAsyncClient	software.amazon.awssdk.services.efs.EfsAsyncClient
com.amazonaws.services.elasticfilesystem.AmazonElasticFileSystemClient	software.amazon.awssdk.services.efs.EfsClient
com.amazonaws.services.elasticloadbalancing.AmazonElasticLoadBalancingAsyncClient	software.amazon.awssdk.services.elasticloadbalancing.ElasticLoadBalancingAsyncClient
com.amazonaws.services.elasticloadbalancing.AmazonElasticLoadBalancingClient	software.amazon.awssdk.services.elasticloadbalancing.ElasticLoadBalancingClient

Client 1.x	Client 2.x
com.amazonaws.services.elasticloadbalancingv2.AmazonElasticLoadBalancingAsyncClient	software.amazon.awssdk.services.elasticloadbalancingv2.ElasticLoadBalancingV2AsyncClient
com.amazonaws.services.elasticloadbalancingv2.AmazonElasticLoadBalancingClient	software.amazon.awssdk.services.elasticloadbalancingv2.ElasticLoadBalancingV2Client
com.amazonaws.services.elasticmapreduce.AmazonElasticMapReduceAsyncClient	software.amazon.awssdk.services.emr.EmrAsyncClient
com.amazonaws.services.elasticmapreduce.AmazonElasticMapReduceClient	software.amazon.awssdk.services.emr.EmrClient
com.amazonaws.services.elasticsearch.AWSElasticsearchAsyncClient	software.amazon.awssdk.services.elasticsearch.ElasticsearchAsyncClient
com.amazonaws.services.elasticsearch.AWSElasticsearchClient	software.amazon.awssdk.services.elasticsearch.ElasticsearchClient
com.amazonaws.services.elastictranscoder.AmazonElasticTranscoderAsyncClient	software.amazon.awssdk.services.elastictranscoder.ElasticTranscoderAsyncClient
com.amazonaws.services.elastictranscoder.AmazonElasticTranscoderClient	software.amazon.awssdk.services.elastictranscoder.ElasticTranscoderClient
com.amazonaws.services.fms.AWSFMSAsyncClient	software.amazon.awssdk.services.fms.FmsAsyncClient

Client 1.x	Client 2.x
com.amazonaws.services.fms.AWSFMSClient	software.amazon.awssdk.services.fms.FmsClient
com.amazonaws.services.gamelift.AmazonGameLiftAsyncClient	software.amazon.awssdk.services.gamelift.GameLiftAsyncClient
com.amazonaws.services.gamelift.AmazonGameLiftClient	software.amazon.awssdk.services.gamelift.GameLiftClient
com.amazonaws.services.glacier.AmazonGlacierAsyncClient	software.amazon.awssdk.services.glacier.GlacierAsyncClient
com.amazonaws.services.glacier.AmazonGlacierClient	software.amazon.awssdk.services.glacier.GlacierClient
com.amazonaws.services.glue.AWSGlueAsyncClient	software.amazon.awssdk.services.glue.GlueAsyncClient
com.amazonaws.services.glue.AWSGlueClient	software.amazon.awssdk.services.glue.GlueClient
com.amazonaws.services.greengrass.AWSGreengrassAsyncClient	software.amazon.awssdk.services.greengrass.GreengrassAsyncClient
com.amazonaws.services.greengrass.AWSGreengrassClient	software.amazon.awssdk.services.greengrass.GreengrassClient
com.amazonaws.services.guardduty.AmazonGuardDutyAsyncClient	software.amazon.awssdk.services.guardduty.GuardDutyAsyncClient
com.amazonaws.services.guardduty.AmazonGuardDutyClient	software.amazon.awssdk.services.guardduty.GuardDutyClient

Client 1.x	Client 2.x
com.amazonaws.services.health.AWSHealthAsyncClient	software.amazon.awssdk.services.health.HealthAsyncClient
com.amazonaws.services.health.AWSHealthClient	software.amazon.awssdk.services.health.HealthClient
com.amazonaws.services.identitymanagement.AmazonIdentityManagementAsyncClient	software.amazon.awssdk.services.iam.IamAsyncClient
com.amazonaws.services.identitymanagement.AmazonIdentityManagementClient	software.amazon.awssdk.services.iam.IamClient
com.amazonaws.services.importexport.AmazonImportExportAsyncClient	*Deprecated*
com.amazonaws.services.importexport.AmazonImportExportClient	*Deprecated*
com.amazonaws.services.inspector.AmazonInspectorAsyncClient	software.amazon.awssdk.services.inspector.InspectorAsyncClient
com.amazonaws.services.inspector.AmazonInspectorClient	software.amazon.awssdk.services.inspector.InspectorClient
com.amazonaws.services.iot.AWSIotAsyncClient	software.amazon.awssdk.services.iot.IotAsyncClient
com.amazonaws.services.iot.AWSIotClient	software.amazon.awssdk.services.iot.IotClient

Client 1.x	Client 2.x
com.amazonaws.services.iot1clickdevices.AWSIoT1ClickDevicesAsyncClient	software.amazon.awssdk.services.iot1clickdevices.Iot1ClickDevicesAsyncClient
com.amazonaws.services.iot1clickdevices.AWSIoT1ClickDevicesClient	software.amazon.awssdk.services.iot1clickdevices.Iot1ClickDevicesClient
com.amazonaws.services.iot1clickprojects.AWSIoT1ClickProjectsAsyncClient	software.amazon.awssdk.services.iot1clickprojects.Iot1ClickProjectsAsyncClient
com.amazonaws.services.iot1clickprojects.AWSIoT1ClickProjectsClient	software.amazon.awssdk.services.iot1clickprojects.Iot1ClickProjectsClient
com.amazonaws.services.iotanalytics.AWSIoTAnalyticsAsyncClient	software.amazon.awssdk.services.iotanalytics.IotAnalyticsAsyncClient
com.amazonaws.services.iotanalytics.AWSIoTAnalyticsClient	software.amazon.awssdk.services.iotanalytics.IotAnalyticsClient
com.amazonaws.services.iotdata.AWSIotDataAsyncClient	software.amazon.awssdk.services.iotdata.IotDataAsyncClient
com.amazonaws.services.iotdata.AWSIotDataClient	software.amazon.awssdk.services.iotdata.IotDataClient
com.amazonaws.services.iotjobsdataplane.AWSIoTJobsDataPlaneAsyncClient	software.amazon.awssdk.services.iotdataplane.IotDataPlaneAsyncClient
com.amazonaws.services.iotjobsdataplane.AWSIoTJobsDataPlaneClient	software.amazon.awssdk.services.iotdataplane.IotDataPlaneClient

Client 1.x	Client 2.x
com.amazonaws.services.kinesis.AmazonKinesisAsyncClient	software.amazon.awssdk.services.kinesis.KinesisAsyncClient
com.amazonaws.services.kinesis.AmazonKinesisClient	software.amazon.awssdk.services.kinesis.KinesisClient
com.amazonaws.services.kinesisanalytics.AmazonKinesisAnalyticsAsyncClient	software.amazon.awssdk.services.kinesisanalytics.KinesisAnalyticsAsyncClient
com.amazonaws.services.kinesisanalytics.AmazonKinesisAnalyticsClient	software.amazon.awssdk.services.kinesisanalytics.KinesisAnalyticsClient
com.amazonaws.services.kinesisfirehose.AmazonKinesisFirehoseAsyncClient	software.amazon.awssdk.services.firehose.FirehoseAsyncClient
com.amazonaws.services.kinesisfirehose.AmazonKinesisFirehoseClient	software.amazon.awssdk.services.firehose.FirehoseClient
com.amazonaws.services.kinesisvideo.AmazonKinesisVideoArchivedMediaAsyncClient	software.amazon.awssdk.services.kinesisvideoarchivedmedia.KinesisVideoArchivedMediaAsyncClient
com.amazonaws.services.kinesisvideo.AmazonKinesisVideoArchivedMediaClient	software.amazon.awssdk.services.kinesisvideoarchivedmedia.KinesisVideoArchivedMediaClient
com.amazonaws.services.kinesisvideo.AmazonKinesisVideoAsyncClient	software.amazon.awssdk.services.kinesisvideo.KinesisVideoAsyncClient

Client 1.x	Client 2.x
`com.amazonaws.services.kinesisvideo.AmazonKinesisVideoClient`	`software.amazon.awssdk.services.kinesisvideo.KinesisVideoClient`
`com.amazonaws.services.kinesisvideo.AmazonKinesisVideoMediaAsyncClient`	`software.amazon.awssdk.services.kinesisvideomedia.KinesisVideoMediaAsyncClient`
`com.amazonaws.services.kinesisvideo.AmazonKinesisVideoMediaClient`	`software.amazon.awssdk.services.kinesisvideomedia.KinesisVideoMediaClient`
`com.amazonaws.services.kinesisvideo.AmazonKinesisVideoPutMediaClient`	Non supportato
`com.amazonaws.services.kms.AWSKMSAsyncClient`	`software.amazon.awssdk.services.kms.KmsAsyncClient`
`com.amazonaws.services.kms.AWSKMSClient`	`software.amazon.awssdk.services.kms.KmsClient`
`com.amazonaws.services.lambda.AWSLambdaAsyncClient`	`software.amazon.awssdk.services.lambda.LambdaAsyncClient`
`com.amazonaws.services.lambda.AWSLambdaClient`	`software.amazon.awssdk.services.lambda.LambdaClient`
`com.amazonaws.services.lexmodelbuilding.AmazonLexModelBuildingAsyncClient`	`software.amazon.awssdk.services.lexmodelbuilding.LexModelBuildingAsyncClient`
`com.amazonaws.services.lexmodelbuilding.AmazonLexModelBuildingClient`	`software.amazon.awssdk.services.lexmodelbuilding.LexModelBuildingClient`

Client 1.x	Client 2.x
com.amazonaws.services.lexruntime.AmazonLexRuntimeAsyncClient	software.amazon.awssdk.services.lexruntime.LexRuntimeAsyncClient
com.amazonaws.services.lexruntime.AmazonLexRuntimeClient	software.amazon.awssdk.services.lexruntime.LexRuntimeClient
com.amazonaws.services.lightsail.AmazonLightsailAsyncClient	software.amazon.awssdk.services.lightsail.LightsailAsyncClient
com.amazonaws.services.lightsail.AmazonLightsailClient	software.amazon.awssdk.services.lightsail.LightsailClient
com.amazonaws.services.logs.AWSLogsAsyncClient	software.amazon.awssdk.services.logs.LogsAsyncClient
com.amazonaws.services.logs.AWSLogsClient	software.amazon.awssdk.services.logs.LogsClient
com.amazonaws.services.machinelearning.AmazonMachineLearningAsyncClient	software.amazon.awssdk.services.machinelearning.MachineLearningAsyncClient
com.amazonaws.services.machinelearning.AmazonMachineLearningClient	software.amazon.awssdk.services.machinelearning.MachineLearningClient
com.amazonaws.services.macie.AmazonMacieAsyncClient	software.amazon.awssdk.services.macie.MacieAsyncClient
com.amazonaws.services.macie.AmazonMacieClient	software.amazon.awssdk.services.macie.MacieClient

Client 1.x	Client 2.x
com.amazonaws.services.marketplacecommerceanalytics.AWSMarketplaceCommerceAnalyticsAsyncClient	software.amazon.awssdk.services.marketplacecommerceanalytics.MarketplaceCommerceAnalyticsAsyncClient
com.amazonaws.services.marketplacecommerceanalytics.AWSMarketplaceCommerceAnalyticsClient	software.amazon.awssdk.services.marketplacecommerceanalytics.MarketplaceCommerceAnalyticsClient
com.amazonaws.services.marketplaceentitlement.AWSMarketplaceEntitlementAsyncClient	software.amazon.awssdk.services.marketplaceentitlement.MarketplaceEntitlementAsyncClient
com.amazonaws.services.marketplaceentitlement.AWSMarketplaceEntitlementClient	software.amazon.awssdk.services.marketplaceentitlement.MarketplaceEntitlementClient
com.amazonaws.services.marketplacemetering.AWSMarketplaceMeteringAsyncClient	software.amazon.awssdk.services.marketplacemetering.MarketplaceMeteringAsyncClient
com.amazonaws.services.marketplacemetering.AWSMarketplaceMeteringClient	software.amazon.awssdk.services.marketplacemetering.MarketplaceMeteringClient
com.amazonaws.services.mediaconvert.AWSMediaConvertAsyncClient	software.amazon.awssdk.services.mediaconvert.MediaConvertAsyncClient
com.amazonaws.services.mediaconvert.AWSMediaConvertClient	software.amazon.awssdk.services.mediaconvert.MediaConvertClient

Client 1.x	Client 2.x
com.amazonaws.services.medialive.AWSMediaLiveAsyncClient	software.amazon.awssdk.services.medialive.MediaLiveAsyncClient
com.amazonaws.services.medialive.AWSMediaLiveClient	software.amazon.awssdk.services.medialive.MediaLiveClient
com.amazonaws.services.mediapackage.AWSMediaPackageAsyncClient	software.amazon.awssdk.services.mediapackage.MediaPackageAsyncClient
com.amazonaws.services.mediapackage.AWSMediaPackageClient	software.amazon.awssdk.services.mediapackage.MediaPackageClient
com.amazonaws.services.mediastore.AWSMediaStoreAsyncClient	software.amazon.awssdk.services.mediastore.MediaStoreAsyncClient
com.amazonaws.services.mediastore.AWSMediaStoreClient	software.amazon.awssdk.services.mediastore.MediaStoreClient
com.amazonaws.services.mediastoredata.AWSMediaStoreDataAsyncClient	software.amazon.awssdk.services.mediastoredata.MediaStoreDataAsyncClient
com.amazonaws.services.mediastoredata.AWSMediaStoreDataClient	software.amazon.awssdk.services.mediastoredata.MediaStoreDataClient
com.amazonaws.services.mediatailor.AWSMediaTailorAsyncClient	software.amazon.awssdk.services.mediatailor.MediaTailorAsyncClient

Client 1.x	Client 2.x
com.amazonaws.services.mediatailor.AWSMediaTailorClient	software.amazon.awssdk.services.mediatailor.MediaTailorClient
com.amazonaws.services.migrationhub.AWSMigrationHubAsyncClient	software.amazon.awssdk.services.migrationhub.MigrationHubAsyncClient
com.amazonaws.services.migrationhub.AWSMigrationHubClient	software.amazon.awssdk.services.migrationhub.MigrationHubClient
com.amazonaws.services.mobile.AWSMobileAsyncClient	software.amazon.awssdk.services.mobile.MobileAsyncClient
com.amazonaws.services.mobile.AWSMobileClient	software.amazon.awssdk.services.mobile.MobileClient
com.amazonaws.services.mq.AmazonMQAsyncClient	software.amazon.awssdk.services.mq.MqAsyncClient
com.amazonaws.services.mq.AmazonMQClient	software.amazon.awssdk.services.mq.MqClient
com.amazonaws.services.mturk.AmazonMTurkAsyncClient	software.amazon.awssdk.services.mturk.MTurkAsyncClient
com.amazonaws.services.mturk.AmazonMTurkClient	software.amazon.awssdk.services.mturk.MTurkClient
com.amazonaws.services.neptune.AmazonNeptuneAsyncClient	software.amazon.awssdk.services.neptune.NeptuneAsyncClient
com.amazonaws.services.neptune.AmazonNeptuneClient	software.amazon.awssdk.services.neptune.NeptuneClient

Client 1.x	Client 2.x
com.amazonaws.services.opsworks.AWSOpsWorksAsyncClient	software.amazon.awssdk.services.opsworks.OpsWorksAsyncClient
com.amazonaws.services.opsworks.AWSOpsWorksClient	software.amazon.awssdk.services.opsworks.OpsWorksClient
com.amazonaws.services.opsworkscm.AWSOpsWorksCMAsyncClient	software.amazon.awssdk.services.opsworkscm.OpsWorksCmAsyncClient
com.amazonaws.services.opsworkscm.AWSOpsWorksCMClient	software.amazon.awssdk.services.opsworkscm.OpsWorksCmClient
com.amazonaws.services.organizations.AWSOrganizationsAsyncClient	software.amazon.awssdk.services.organizations.OrganizationsAsyncClient
com.amazonaws.services.organizations.AWSOrganizationsClient	software.amazon.awssdk.services.organizations.OrganizationsClient
com.amazonaws.services.pi.AWSPIAsyncClient	software.amazon.awssdk.services.pi.PiAsyncClient
com.amazonaws.services.pi.AWSPIClient	software.amazon.awssdk.services.pi.PiClient
com.amazonaws.services.pinpoint.AmazonPinpointAsyncClient	software.amazon.awssdk.services.pinpoint.PinpointAsyncClient
com.amazonaws.services.pinpoint.AmazonPinpointClient	software.amazon.awssdk.services.pinpoint.PinpointClient

Client 1.x	Client 2.x
com.amazonaws.services.polly.AmazonPollyAsyncClient	software.amazon.awssdk.services.polly.PollyAsyncClient
com.amazonaws.services.polly.AmazonPollyClient	software.amazon.awssdk.services.polly.PollyClient
com.amazonaws.services.pricing.AWSPricingAsyncClient	software.amazon.awssdk.services.pricing.PricingAsyncClient
com.amazonaws.services.pricing.AWSPricingClient	software.amazon.awssdk.services.pricing.PricingClient
com.amazonaws.services.rds.AmazonRDSAsyncClient	software.amazon.awssdk.services.rds.RdsAsyncClient
com.amazonaws.services.rds.AmazonRDSClient	software.amazon.awssdk.services.rds.RdsClient
com.amazonaws.services.redshift.AmazonRedshiftAsyncClient	software.amazon.awssdk.services.redshift.RedshiftAsyncClient
com.amazonaws.services.redshift.AmazonRedshiftClient	software.amazon.awssdk.services.redshift.RedshiftClient
com.amazonaws.services.rekognition.AmazonRekognitionAsyncClient	software.amazon.awssdk.services.rekognition.RekognitionAsyncClient
com.amazonaws.services.rekognition.AmazonRekognitionClient	software.amazon.awssdk.services.rekognition.RekognitionClient
com.amazonaws.services.resourcegroups.AWSResourceGroupsAsyncClient	software.amazon.awssdk.services.resourcegroups.ResourceGroupsAsyncClient

Client 1.x	Client 2.x
com.amazonaws.services.resourcegroups.AWSResourceGroupsClient	software.amazon.awssdk.services.resourcegroups.ResourceGroupsClient
com.amazonaws.services.resourcegroupstaggingapi.AWSResourceGroupsTaggingAPIAsyncClient	software.amazon.awssdk.services.resourcegroupstaggingapi.ResourceGroupsTaggingApiAsyncClient
com.amazonaws.services.resourcegroupstaggingapi.AWSResourceGroupsTaggingAPIClient	software.amazon.awssdk.services.resourcegroupstaggingapi.ResourceGroupsTaggingApiClient
com.amazonaws.services.route53.AmazonRoute53AsyncClient	software.amazon.awssdk.services.route53.Route53AsyncClient
com.amazonaws.services.route53.AmazonRoute53Client	software.amazon.awssdk.services.route53.Route53Client
com.amazonaws.services.route53domains.AmazonRoute53DomainsAsyncClient	software.amazon.awssdk.services.route53domains.Route53DomainsAsyncClient
com.amazonaws.services.route53domains.AmazonRoute53DomainsClient	software.amazon.awssdk.services.route53domains.Route53DomainsClient
com.amazonaws.services.s3.AmazonS3Client	software.amazon.awssdk.services.s3.S3Client
com.amazonaws.services.sagemaker.AmazonSageMakerAsyncClient	software.amazon.awssdk.services.sagemaker.SageMakerAsyncClient
com.amazonaws.services.sagemaker.AmazonSageMakerClient	software.amazon.awssdk.services.sagemaker.SageMakerClient

Client 1.x	Client 2.x
com.amazonaws.services.sagemakerruntime.AmazonSageMakerRuntimeAsyncClient	software.amazon.awssdk.services.sagemakerruntime.SageMakerRuntimeAsyncClient
com.amazonaws.services.sagemakerruntime.AmazonSageMakerRuntimeClient	software.amazon.awssdk.services.sagemakerruntime.SageMakerRuntimeClient
com.amazonaws.services.secretsmanager.AWSSecretsManagerAsyncClient	software.amazon.awssdk.services.secretsmanager.SecretsManagerAsyncClient
com.amazonaws.services.secretsmanager.AWSSecretsManagerClient	software.amazon.awssdk.services.secretsmanager.SecretsManagerClient
com.amazonaws.services.securitytoken.AWSSecurityTokenServiceAsyncClient	software.amazon.awssdk.services.sts.StsAsyncClient
com.amazonaws.services.securitytoken.AWSSecurityTokenServiceClient	software.amazon.awssdk.services.sts.StsClient
com.amazonaws.services.serverlessapplicationrepository.AWSServerlessApplicationRepositoryAsyncClient	software.amazon.awssdk.services.serverlessapplicationrepository.ServerlessApplicationRepositoryAsyncClient
com.amazonaws.services.serverlessapplicationrepository.AWSServerlessApplicationRepositoryClient	software.amazon.awssdk.services.serverlessapplicationrepository.ServerlessApplicationRepositoryClient

Client 1.x	Client 2.x
com.amazonaws.services.servermigration.AWSServerMigrationAsyncClient	software.amazon.awssdk.services.sms.SmsAsyncClient
com.amazonaws.services.servermigration.AWSServerMigrationClient	software.amazon.awssdk.services.sms.SmsClient
com.amazonaws.services.servicecatalog.AWSServiceCatalogAsyncClient	software.amazon.awssdk.services.servicecatalog.ServiceCatalogAsyncClient
com.amazonaws.services.servicecatalog.AWSServiceCatalogClient	software.amazon.awssdk.services.servicecatalog.ServiceCatalogClient
com.amazonaws.services.servicediscovery.AWSServiceDiscoveryAsyncClient	software.amazon.awssdk.services.servicediscovery.ServiceDiscoveryAsyncClient
com.amazonaws.services.servicediscovery.AWSServiceDiscoveryClient	software.amazon.awssdk.services.servicediscovery.ServiceDiscoveryClient
com.amazonaws.services.shield.AWSShieldAsyncClient	software.amazon.awssdk.services.shield.ShieldAsyncClient
com.amazonaws.services.shield.AWSShieldClient	software.amazon.awssdk.services.shield.ShieldClient
com.amazonaws.services.simpledb.AmazonSimpleDBAsyncClient	software.amazon.awssdk.services.simpledb.SimpleDbAsyncClient
com.amazonaws.services.simpledb.AmazonSimpleDBClient	software.amazon.awssdk.services.simpledb.SimpleDbClient

Client 1.x	Client 2.x
`com.amazonaws.services.simpleemail.AmazonSimpleEmailServiceAsyncClient`	`software.amazon.awssdk.services.ses.SesAsyncClient`
`com.amazonaws.services.simpleemail.AmazonSimpleEmailServiceClient`	`software.amazon.awssdk.services.ses.SesClient`
`com.amazonaws.services.simplesystemsmanagement.AWSSimpleSystemsManagementAsyncClient`	`software.amazon.awssdk.services.ssm.SsmAsyncClient`
`com.amazonaws.services.simplesystemsmanagement.AWSSimpleSystemsManagementClient`	`software.amazon.awssdk.services.ssm.SsmClient`
`com.amazonaws.services.simpleworkflow.AmazonSimpleWorkflowAsyncClient`	`software.amazon.awssdk.services.swf.SwfAsyncClient`
`com.amazonaws.services.simpleworkflow.AmazonSimpleWorkflowClient`	`software.amazon.awssdk.services.swf.SwfClient`
`com.amazonaws.services.snowball.AmazonSnowballAsyncClient`	`software.amazon.awssdk.services.snowball.SnowballAsyncClient`
`com.amazonaws.services.snowball.AmazonSnowballClient`	`software.amazon.awssdk.services.snowball.SnowballClient`
`com.amazonaws.services.sns.AmazonSNSAsyncClient`	`software.amazon.awssdk.services.sns.SnsAsyncClient`
`com.amazonaws.services.sns.AmazonSNSClient`	`software.amazon.awssdk.services.sns.SnsClient`

Client 1.x	Client 2.x
com.amazonaws.services.sqs.AmazonSQSAsyncClient	software.amazon.awssdk.services.sqs.SqsAsyncClient
com.amazonaws.services.sqs.AmazonSQSClient	software.amazon.awssdk.services.sqs.SqsClient
com.amazonaws.services.stepfunctions.AWSStepFunctionsAsyncClient	software.amazon.awssdk.services.sfn.SfnAsyncClient
com.amazonaws.services.stepfunctions.AWSStepFunctionsClient	software.amazon.awssdk.services.sfn.SfnClient
com.amazonaws.services.storagegateway.AWSStorageGatewayAsyncClient	software.amazon.awssdk.services.storagegateway.StorageGatewayAsyncClient
com.amazonaws.services.storagegateway.AWSStorageGatewayClient	software.amazon.awssdk.services.storagegateway.StorageGatewayClient
com.amazonaws.services.support.AWSSupportAsyncClient	software.amazon.awssdk.services.support.SupportAsyncClient
com.amazonaws.services.support.AWSSupportClient	software.amazon.awssdk.services.support.SupportClient
com.amazonaws.services.transcribe.AmazonTranscribeAsyncClient	software.amazon.awssdk.services.transcribe.TranscribeAsyncClient
com.amazonaws.services.transcribe.AmazonTranscribeClient	software.amazon.awssdk.services.transcribe.TranscribeClient

Client 1.x	Client 2.x
com.amazonaws.services.translate.AmazonTranslateAsyncClient	software.amazon.awssdk.services.translate.TranslateAsyncClient
com.amazonaws.services.translate.AmazonTranslateClient	software.amazon.awssdk.services.translate.TranslateClient
com.amazonaws.services.waf.AWSWAFAsyncClient	software.amazon.awssdk.services.waf.WafAsyncClient
com.amazonaws.services.waf.AWSWAFClient	software.amazon.awssdk.services.waf.WafClient
com.amazonaws.services.waf.AWSWAFRegionalAsyncClient	software.amazon.awssdk.services.waf.regional.WafRegionalAsyncClient
com.amazonaws.services.waf.AWSWAFRegionalClient	software.amazon.awssdk.services.waf.regional.WafRegionalClient
com.amazonaws.services.workdocs.AmazonWorkDocsAsyncClient	software.amazon.awssdk.services.workdocs.WorkDocsAsyncClient
com.amazonaws.services.workdocs.AmazonWorkDocsClient	software.amazon.awssdk.services.workdocs.WorkDocsClient
com.amazonaws.services.workmail.AmazonWorkMailAsyncClient	software.amazon.awssdk.services.workmail.WorkMailAsyncClient
com.amazonaws.services.workmail.AmazonWorkMailClient	software.amazon.awssdk.services.workmail.WorkMailClient

Client 1.x	Client 2.x
`com.amazonaws.services.workspaces.AmazonWorkspacesAsyncClient`	`software.amazon.awssdk.services.workspaces.WorkSpacesAsyncClient`
`com.amazonaws.services.workspaces.AmazonWorkspacesClient`	`software.amazon.awssdk.services.workspaces.WorkSpacesClient`
`com.amazonaws.services.xray.AWSXRayAsyncClient`	`software.amazon.awssdk.services.xray.XRayAsyncClient`
`com.amazonaws.services.xray.AWSXRayClient`	`software.amazon.awssdk.services.xray.XRayClient`

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
`AWS_CBOR_DISABLED`	`CBOR_ENABLED`
`AWS_ION_BINARY_DISABLE`	`BINARY_ION_ENABLED`

- Le seguenti proprietà di sistema relative alla creazione di client predefiniti sono diverse.

1.x	2.x
`com.amazonaws.sdk.disableEc2Metadata`	`aws.disableEc2Metadata`
`com.amazonaws.sdk.ec2MetadataServiceEndpointOverride`	`aws.ec2MetadataServiceEndpoint`
`com.amazonaws.sdk.disableCbor`	`aws.cborEnabled`
`com.amazonaws.sdk.disableIonBinary`	`aws.binaryIonEnabled`

- La versione 2.x non supporta le seguenti proprietà di sistema.
-

1.x
`com.amazonaws.sdk.disableCertChecking`
`com.amazonaws.sdk.enableDefaultMetrics`
`com.amazonaws.sdk.enableThrottledRetry`
`com.amazonaws.regions.RegionUtils.fileOverride`
`com.amazonaws.regions.RegionUtils.disableRemote`
`com.amazonaws.services.s3.disableImplicitGlobalClients`
`com.amazonaws.sdk.enableInRegionOptimizedMode`

- 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
	`ClientConfiguration clientConfig = new ClientConfiguration()`	`ApacheHttpClient.Builder httpClientBuilder = ApacheHttpClient.builder()`	`NettyNioAsyncHttpClient.Builder httpClientBuilder = NettyNioAsyncHttpClient.builder()`
Numero massimo di connessioni	`clientConfig.setMaxConnections(...)` `clientConfig.withMaxConnections(...)`	`httpClientBuilder.maxConnections(...)`	`httpClientBuilder.maxConcurrency(...)`

Impostazione	1.x	Sincronizzazione 2.x, Apache	2.x Asincrono, Netty
Timeout di connessione	`clientConfig.setConnectionTimeout(...)` `clientConfig.withConnectionTimeout(...)`	`httpClientBuilder.connectionTimeout(...)` `httpClientBuilder.connectionAcquisitionTimeout(...)`	`httpClientBuilder.connectionTimeout(...)`
Timeout del socket	`clientConfig.setSocketTimeout(...)` `clientConfig.withSocketTimeout(...)`	`httpClientBuilder.socketTimeout(...)`	`httpClientBuilder.writeTimeout(...)` `httpClientBuilder.readTimeout(...)`
Connessione TTL	`clientConfig.setConnectionTTL(...)` `clientConfig.withConnectionTTL(...)`	`httpClientBuilder.connectionTimeToLive(...)`	`httpClientBuilder.connectionTimeToLive(...)`

Impostazione	1.x	Sincronizzazione 2.x, Apache	2.x Asincrono, Netty
Connessione massima inattiva	`clientConfig.setConnectionMaxIdleMillis(...)` `clientConfig.withConnectionMaxIdleMillis(...)`	`httpClientBuilder.connectionMaxIdleTime(...)`	`httpClientBuilder.connectionMaxIdleTime(...)`
Convalida dopo l'inattività	`clientConfig.setValidateAfterInactivityMillis(...)` `clientConfig.withValidateAfterInactivityMillis(...)`	Non supportato (funzionalità di richiesta)	Non supportato (funzionalità di richiesta)
Indirizzo locale	`clientConfig.setLocalAddress(...)` `clientConfig.withLocalAddress(...)`	`httpClientBuilder.localAddress(...)`	Non supportato
Expect-continue abilitato	`clientConfig.setUseExpectContinue(...)` `clientConfig.withUseExpectContinue(...)`	`httpClientBuilder.expectContinueEnabled(...)`	Non supportato (funzionalità di richiesta)

Impostazione	1.x	Sincronizzazione 2.x, Apache	2.x Asincrono, Netty
Connection Reaper	`clientConfig.setUseReaper(...)` `clientConfig.withReaper(...)`	`httpClientBuilder.useIdleConnectionReaper(...)`	`httpClientBuilder.useIdleConnectionReaper(...)`
	`AmazonDynamoDBClientBuilder.standard()` `.withClientConfiguration(clientConfiguration)` `.build()`	`DynamoDbClient.builder()` `.httpClientBuilder(httpClientBuilder)` `.build()`	`DynamoDbAsyncClient.builder()` `.httpClientBuilder(httpClientBuilder)` `.build()`

Proxy client HTTP

Impostazioni	1.x	Sincronizzazione 2.x, Apache	2.x Asincrono, Netty
	`ClientConfiguration clientConfig = new ClientConfiguration()`	`ProxyConfiguration.Builder proxyConfig = ProxyConfiguration.builder()`	`ProxyConfiguration.Builder proxyConfig = ProxyConfiguration.builder()`
Host proxy	`clientConfig.setProxyHost(...)`	`proxyConfig.endpoint(...)`	`proxyConfig.host(...)`

Impostazioni	1.x	Sincronizzazione 2.x, Apache	2.x Asincrono, Netty
	`clientConfig.withProxyHost(...)`		
Porta proxy	`clientConfig.setProxyPort(...)` `clientConfig.withProxyPort(...)`	`proxyConfig.endpoint(...)` La porta proxy è incorporata in `endpoint`	`proxyConfig.port(...)`
Nome utente proxy	`clientConfig.setProxyUsername(...)` `clientConfig.withProxyUsername(...)`	`proxyConfig.username(...)`	`proxyConfig.username(...)`
Password proxy	`clientConfig.setProxyPassword(...)` `clientConfig.withProxyPassword(...)`	`proxyConfig.password(...)`	`proxyConfig.password(...)`
Dominio proxy	`clientConfig.setProxyDomain(...)` `clientConfig.withProxyDomain(...)`	`proxyConfig.ntlmDomain(...)`	Non supportato (funzionalità di richiesta)

Impostazioni	1.x	Sincronizzazione 2.x, Apache	2.x Asincrono, Netty
Workstation proxy	`clientConfig.setProxyWorkspace(...)` `clientConfig.withProxyWorkstation(...)`	`proxyConfig.ntlmWorkstation(...)`	Non supportata (funzionalità di richiesta)
Metodi di autenticazione proxy	`clientConfig.setProxyAuthenticationMethods(...)` `clientConfig.withProxyAuthenticationMethods(...)`	Non supportato	Non supportato (funzionalità di richiesta)
Autenticazione proxy di base preventiva	`clientConfig.setPreemptiveBasicProxyAuth(...)` `clientConfig.withPreemptiveBasicProxyAuth(...)`	`proxyConfig.preemptiveBasicAuthenticationEnabled(...)`	Non supportata (funzionalità di richiesta)

Impostazioni	1.x	Sincronizzazione 2.x, Apache	2.x Asincrono, Netty
Host non proxy	`clientConfig.setNonProxyHosts(...)` `clientConfig.withNonProxyHosts(...)`	`proxyConfig.nonProxyHosts(...)`	`proxyConfig.nonProxyHosts(...)`
Disabilita il socket proxy	`clientConfig.setDisableSocketProxy(...)` `clientConfig.withDisableSocketProxy(...)`	Non supportato (funzionalità di richiesta)	Non supportato (funzionalità di richiesta)

	`AmazonDynamoDBClientBuilder.standard()`
	`.withClientConfiguration(clientConfiguration)`
	`.build()`
		`httpClientBuilder.proxyConfiguration(`
		`proxyConfig.build())`
			`httpClientBuilder.proxyConfiguration(`
			`proxyConfig.build())`

Sostituzioni del client

Impostazione	1.x	2.x
	`ClientConfiguration clientConfig = new ClientConfiguration()`	`ClientOverrideConfiguration.Builder overrideConfig = ClientOverrideConfiguration.builder()`
Prefisso dell'agente utente	`clientConfig.setUserAgentPrefix(...)` `clientConfig.withUserAgentPrefix(...)`	`overrideConfig.advancedOption( SdkAdvancedClientOption.USER_AGENT_PREFIX, ...)`
Suffisso dell'agente utente	`clientConfig.setUserAgentSuffix(...)` `clientConfig.withUserAgentSuffix(...)`	`overrideConfig.advancedOption( SdkAdvancedClientOption.USER_AGENT_SUFFIX, ...)`
Signer	`clientConfig.setSignerOverride(...)` `clientConfig.withSignerOverride(...)`	`overrideConfig.advancedOption( SdkAdvancedClientOption.SIGNER, ...)`
Intestazioni aggiuntive	`clientConfig.addHeader(...)` `clientConfig.withHeader(...)`	`overrideConfig.putHeader(...)`
Timeout richiesta	`clientConfig.setRequestTimeout(...)` `clientConfig.withRequestTimeout(...)`	`overrideConfig.apiCallAttemptTimeout(...)`

Impostazione	1.x	2.x
Timeout di esecuzione del client	`clientConfig.setClientExecutionTimeout(...)` `clientConfig.withClientExecutionTimeout(...)`	`overrideConfig.apiCallTimeout(...)`
Usa Gzip	`clientConfig.setUseGzip(...)` `clientConfig.withGzip(...)`	Non supportato (funzionalità di richiesta)
Suggerimento sulla dimensione del buffer del socket	`clientConfig.setSocketBufferSizeHints(...)` `clientConfig.withSocketBufferSizeHints(...)`	Non supportato (funzionalità di richiesta)
Metadati di risposta alla cache	`clientConfig.setCacheResponseMetadata(...)` `clientConfig.withCacheResponseMetadata(...)`	Non supportato (funzionalità di richiesta)
Dimensione della cache dei metadati di risposta	`clientConfig.setResponseMetadataCacheSize(...)` `clientConfig.withResponseMetadataCacheSize(...)`	Non supportata (funzionalità di richiesta)

Impostazione	1.x	2.x
resolver DNS	`clientConfig.setDnsResolver(...)` `clientConfig.withDnsResolver(...)`	Non supportato (funzionalità di richiesta)

TCP keepalive	`clientConfig.setUseTcpKeepAlive(...)`
	`clientConfig.withTcpKeepAlive(...)`
		Questa opzione è ora nella configurazione del client HTTP
		`- ApacheHttpClient.builder().tcpKeepAlive(true)`
		`- NettyNioAsyncHttpClient.builder().tcpKeepAlive(true)`

Secure Random	`clientConfig.setSecureRandom(...)` `clientConfig.withSecureRandom(...)`	Non supportato (funzionalità di richiesta)

	`AmazonDynamoDBClientBuilder.standard()`
	`.withClientConfiguration(clientConfiguration)`
	`.build()`
		`DynamoDbClient.builder()`
		`.overrideConfiguration(overrideConfig.build())`
		`.build()`

Il client annulla i tentativi

Impostazione	1.x	2.x
	`ClientConfiguration clientConfig =` `new ClientConfiguration()`	`ClientOverrideConfiguration.Builder overrideConfigBuilder =`

Impostazione	1.x	2.x
		`ClientOverrideConfiguration.builder();`
Numero massimo di tentativi di errore	`clientConfig.setMaxErrorRetry(...)` `clientConfig.withMaxErrorRetry(...)`	`// Configure the default retry strategy.` `overrideConfigBuilder.retryStrategy(b -> b.maxAttempts(...));`
Usa tentativi limitati	`clientConfig.setUseThrottleRetries(...)` `clientConfig.withUseThrottleRetries(...)`	Non supportato
Numero massimo di tentativi consecutivi prima della limitazione	`clientConfig.setMaxConsecutiveRetriesBeforeThrottling(...)` `clientConfig.withMaxConsecutiveRetriesBeforeThrottling(...)`	Non supportato

Impostazione	1.x	2.x

	`AmazonDynamoDBClientBuilder.standard()`
	`    .withClientConfiguration(clientConfiguration)`
	`    .build()`
		`DynamoDbClient.builder()`
		`        .overrideConfiguration(overrideConfigBuilder.build())`
		`        .build();`
		
		`// You also have the option to use a lambda expression to configure and`
		`// build the 'ClientOverrideConfiguration.Builder'.`
		`DynamoDbClient client = DynamoDbClient.builder()`
		`        .overrideConfiguration(o -> o.retryStrategy(b -> b.maxAttempts(5)))`
		`        .build();`

Client asincroni

Impostazione	1.x	2.x
		`ClientAsyncConfiguration.Builder asyncConfig = ClientAsyncConfiguration.builder()`
Esecutore	`AmazonDynamoDBAsyncClientBuilder.standard()`	`asyncConfig.advancedOption(`

Impostazione	1.x	2.x
	`.withExecutorFactory(...)` `.build()`	`SdkAdvancedAsyncClientOption.FUTURE_COMPLETION_EXECUTOR, ...)`

		`DynamoDbAsyncClient.builder()`
		`.asyncConfiguration(asyncConfig)`
		`.build()`

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	`clientConfig.setProtocol(Protocol.HTTP)`
	`clientConfig.withProtocol(Protocol.HTTP)`
		L'impostazione predefinita del protocollo è HTTPS. Per modificare l'impostazione, specifica il protocollo che imposta un endpoint HTTP nel client builder:
		`clientBuilder.endpointOverride(`
		`URI.create("http://..."))`

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 per 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 utente.

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	`com.amazonaws.auth.AWSCredentialsProvider`	`software.amazon.awssdk.auth.credentials.AwsCredentialsProvider`
Nome del metodo	`getCredentials`	`resolveCredentials`
Metodo non supportato	`refresh`	Non supportato

`DefaultAWSCredentialsProviderChain`

Cambia categoria	1.x	2.x
Nome del pacchetto/classe	`com.amazonaws.auth.DefaultAWSCredentialsProviderChain`	`software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider`
Creazione	`new DefaultAWSCredentialsProviderChain`	`DefaultCredentialsProvider.create`
Metodo non supportato	`getInstance`	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	`com.amazonaws.auth.AWSStaticCredentialsProvider`	`software.amazon.awssdk.auth.credentials.StaticCredentialsProvider`
Creazione	`new AWSStaticCredentialsProvider`	`StaticCredentialsProvider.create`

`EnvironmentVariableCredentialsProvider`

Cambia categoria	1.x	2.x
Nome del pacchetto/classe	`com.amazonaws.auth.EnvironmentVariableCredentialsProvider`	`software.amazon.awssdk.auth.credentials.EnvironmentVariableCredentialsProvider`
Creazione	`new EnvironmentVariableCredentialsProvider`	`EnvironmentVariableCredentialsProvider.create`
Nome della variabile di ambiente	`AWS_ACCESS_KEY`	`AWS_ACCESS_KEY_ID`
	`AWS_SECRET_KEY`	`AWS_SECRET_ACCESS_KEY`

`SystemPropertiesCredentialsProvider`

Cambia categoria	1.x	2.x
Nome del pacchetto/classe	`com.amazonaws.auth.SystemPropertiesCredentialsProvider`	`software.amazon.awssdk.auth.credentials.SystemPropertyCredentialsProvider`
Creazione	`new SystemPropertiesCredentialsProvider`	`SystemPropertiesCredentialsProvider.create`
Nome della proprietà del sistema	`aws.secretKey`	`aws.secretAccessKey`

`ProfileCredentialsProvider`

Cambia categoria	1.x	2.x
Nome del pacchetto/classe	`com.amazonaws.auth.profile.ProfileCredentialsProvider`	`software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider`
Creazione	`new ProfileCredentialsProvider`	`ProfileCredentialsProvider.create`
Ubicazione del profilo personalizzato	• Variabile di ambiente `AWS_CREDENTIAL_PROFILES_FILE` • `new ProfileCredentialsProvider`	• Variabile di ambiente `AWS_SHARED_CREDENTIALS_FILE` • `ProfileCredentialsProvider.builder`

`ContainerCredentialsProvider`

Cambia categoria	1.x	2.x
Nome del pacchetto/classe	`com.amazonaws.auth.ContainerCredentialsProvider`	`software.amazon.awssdk.auth.credentials.ContainerCredentialsProvider`
Creazione	`new ContainerCredentialsProvider`	`ContainerCredentialsProvider.create`
Specificare l'aggiornamento asincrono	Non supportato	Comportamento predefinito

`InstanceProfileCredentialsProvider`

Cambia categoria	1.x	2.x
Nome del pacchetto/classe	`com.amazonaws.auth.InstanceProfileCredentialsProvider`	`software.amazon.awssdk.auth.credentials.InstanceProfileCredentialsProvider`
Creazione	`new InstanceProfileCredentialsProvider`	`InstanceProfileCredentialsProvider.create`
Specificare l'aggiornamento asincrono	`new InstanceProfileCredentialsProvider(true)`	`InstanceProfileCredentialProvider.builder().asyncCredentialUpdateEnabled(true).build()`
Nome della proprietà di sistema	`com.amazonaws.sdk.disableEc2Metadata`	`aws.disableEc2Metadata`

Cambia categoria	1.x	2.x
	`com.amazonaws.sdk.ec2MetadataServiceEndpointOverride`	`aws.ec2MetadataServiceEndpoint`

`STSAssumeRoleSessionCredentialsProvider`

Cambia categoria	1.x	2.x
Nome del pacchetto/classe	`com.amazonaws.auth.STSAssumeRoleSessionCredentialsProvider`	`software.amazon.awssdk.services.sts.auth.StsAssumeRoleCredentialsProvider`
Creazione	• `new STSAssumeRoleSessionCredentialsProvider` • `new STSAssumeRoleSessionCredentialsProvider.Builder`	`StsAssumeRoleCredentialsProvider.builder`
Aggiornamento asincrono	Comportamento predefinito	Comportamento predefinito
Configurazione	`new STSAssumeRoleSessionCredentialsProvider.Builder`	Configura una richiesta e `StsClient AssumeRoleRequest`

`STSSessionCredentialsProvider`

Cambia categoria	1.x	2.x
Nome del pacchetto/classe	`com.amazonaws.auth.STSSessionCredentialsProvider`	`software.amazon.awssdk.services.sts.auth.StsGetSessionTokenCredentialsProvider`
Creazione	`new STSSessionCredentialsProvider`	`StsGetSessionTokenCredentialsProvider.builder`
Aggiornamento asincrono	Comportamento predefinito	`StsGetSessionTokenCredentialsProvider.builder`
Configurazione	Parametri del costruttore	Configura una `GetSessionTokenRequest` richiesta `StsClient` and in un builder

`WebIdentityFederationSessionCredentialsProvider`

Cambia categoria	1.x	2.x
Nome del pacchetto/classe	`com.amazonaws.auth.WebIdentityFederationSessionCredentialsProvider`	`software.amazon.awssdk.services.sts.auth.StsAssumeRoleWithWebIdentityCredentialsProvider`
Creazione	`new WebIdentityFederationSessionCredentialsProvider`	`StsAssumeRoleWithWebIdentityCredentialsProvider.builder`

Cambia categoria	1.x	2.x
Aggiornamento asincrono	Comportamento predefinito	`StsAssumeRoleWithWebIdentityCredentialsProvider.builder`
Configurazione	Parametri del costruttore	Configura una `AssumeRoleWithWebIdentityRequest` richiesta `StsClient` and in un builder

Classi sostituite

classe 1.x	classi sostitutive 2.x
`com.amazonaws.auth.EC2ContainerCredentialsProviderWrapper`	`software.amazon.awssdk.auth.credentials.ContainerCredentialsProvider` e `software.amazon.awssdk.auth.credentials.InstanceProfileCredentialsProvider`
`com.amazonaws.services.s3.S3CredentialsProviderChain`	`software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider` e `software.amazon.awssdk.auth.credentials.AnonymousCredentialsProvider`

Classi rimosse

classe 1.x
`com.amazonaws.auth.ClasspathPropertiesFileCredentialsProvider`
`com.amazonaws.auth.PropertiesFileCredentialsProvider`

Modifiche alla regione

Questa sezione descrive le modifiche implementate nella versione AWS SDK per 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 per JavaÈ 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
`Regions.fromName`	`Region.of`
`Regions.getName`	`Region.id`
`Regions.getDescription`	`Region.metadata().description()`
`Regions.getCurrentRegion`	Non supportato
`Regions.DEFAULT_REGION`	Non supportato

1.x	2.x
`Regions.name`	`Region.id`

1.x Modifiche al metodo della classe Region

1.x	2.x
`Region.getName`	`Region.id`
`Region.hasHttpsEndpoint`	Non supportato
`Region.hasHttpEndpoint`	Non supportato
`Region.getAvailableEndpoints`	Non supportato
`Region.createClient`	Non supportato

RegionMetadata modifiche al metodo di classe

1.x	2.x
`RegionMetadata.getName`	`RegionMetadata.name`
`RegionMetadata.getDomain`	`RegionMetadata.domain`
`RegionMetadata.getPartition`	`RegionMetadata.partition`

ServiceMetadata modifiche al metodo di classe

1.x	2.x
`Region.getServiceEndpoint`	`ServiceMetadata.endpointFor(Region)`
`Region.isServiceSupported`	`ServiceMetadata.regions().contains(Region)`

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(...))`

Differenze nelle operazioni di streaming tra 1.x e 2.x del AWS SDK per Java

Le operazioni di streaming, come Amazon S3 `getObject` e `putObject` i metodi, supportano l'I/O non bloccante nella versione 2.x dell'SDK. Di conseguenza, gli oggetti del modello di richiesta e risposta non accettano più un parametro come parametro. `InputStream` Per le richieste sincrone, invece, l'oggetto di 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")));
```

Un oggetto di risposta in streaming accetta un `ResponseTransformer` per i client sincroni e un `AsyncResponseTransformer` per i client asincroni nella V2.

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 conversioni di tipo comune in memoria.

Differenze di serializzazione tra 1.x e 2.x del AWS SDK per Java

Elenca gli oggetti per richiedere la differenza dei parametri

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 List`listParam`, 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=
```

POJOs in V1 rispetto ai builder in V2

Poiché l'SDK V1 per Java utilizza classi POJO mutabili, le librerie di serializzazione e deserializzazione, come Jackson, possono utilizzare direttamente gli oggetti del modello.

L'SDK V2 per Java, al contrario, utilizza oggetti del modello immutabili. È necessario utilizzare un builder intermedio per eseguire la de/serializzazione.

L'esempio seguente mostra le differenze tra la de/serializzazione di una chiamata `headBucket` API con V1 e V2 utilizzando un Jackson. `0bjectMapper`

```
    public void sendRequest() throws IOException {
        final String bucketName = "amzn-s3-demo-bucket";
        final ObjectMapper mapper = new ObjectMapper();

        // V1 uses POJOs to serialize and deserialize.
        final AmazonS3 v1S3Client = AmazonS3ClientBuilder.defaultClient();
        HeadBucketResult resultV1 = v1S3Client.headBucket(
            new HeadBucketRequest(bucketName));

        String v1Serialized = mapper.writeValueAsString(resultV1);

        HeadBucketResult deserializedV1 = mapper.readValue(v1Serialized,
 HeadBucketResult.class);

        // V2 uses builders to serialize and deserialize.
        S3Client v2S3Client = S3Client.create();
        HeadBucketResponse v2Response = v2S3Client.headBucket(
            b -> b.bucket(bucketName));

        String v2Serialized = mapper.writeValueAsString(
            v2Response.toBuilder());
```

```
        HeadBucketResponse v2Deserialized = mapper.readValue(
            v2Serialized, HeadBucketResponse.serializableBuilderClass())
            .build();
    }
```

Differenze di deserializzazione tra 1.x e 2.x del AWS SDK per Java

Raccolte vuote nella V2 rispetto alla V1 **`nulls`**

Gli SDK for Java v1.x e v2.x differiscono nel modo in cui deserializzano le risposte JSON con elenchi e mappe vuoti.

Quando l'SDK riceve una risposta con una proprietà mancante modellata come elenco o mappa, V1 deserializza la proprietà mancante su, mentre V2 deserializza la proprietà su un oggetto di raccolta vuoto immutabile. `null`

Ad esempio, considerate la risposta restituita per il `describeTable` metodo da un client DynamoDB. Il seguente metodo di esempio contiene un client DynamoDB V2 e un client DynamoDB V1 che eseguono entrambi il metodo su una tabella che non ha indici secondari globali`describeTable`.

Example della deserializzazione di una proprietà modellata come elenco mancante nella risposta

```
public void deserializationDiffs(){

    DescribeTableResponse v2Response = dynamoDbClientV2.describeTable(builder ->
 builder.tableName(TABLE_NAME));
    // V2 provides has* methods on model objects for list/map members. No null check
 needed.
    LOGGER.info( String.valueOf(v2Response.table().hasGlobalSecondaryIndexes()) );

 LOGGER.info( String.valueOf(v2Response.table().globalSecondaryIndexes().isEmpty()) );
    // V2 deserialize to an empty collection.
    LOGGER.info(v2Response.table().globalSecondaryIndexes().toString());

    // V1 deserialize to null.
    DescribeTableResult v1Result = dynamoDbClientV1.describeTable(new
 DescribeTableRequest(TABLE_NAME));
    if (v1Result.getTable().getGlobalSecondaryIndexes() != null){
        LOGGER.info(v1Result.getTable().getGlobalSecondaryIndexes().toString());
    } else {
```

```
        LOGGER.info("The list of global secondary indexes returned by the V1 call is
 <null>");
    }
}
```

Quanto segue mostra l'output registrato:

```
INFO  org.example.DeserializationDifferences:45 - false
INFO  org.example.DeserializationDifferences:46 - true
INFO  org.example.DeserializationDifferences:48 - []
INFO  org.example.DeserializationDifferences:55 - The list of global secondary indexes
 returned by the V1 call is <null>
```

Java SDK 2.x adotta un approccio idiomatico deserializzando elenchi e mappe vuoti su raccolte vuote immutabili anziché restituirle, promuovendo un codice più sicuro e conciso. `null` Con V2, è possibile verificare se un servizio ha restituito un attributo modellato come elenco o mappa con il metodo, come mostrato nell'esempio precedente. `has*` `hasGlobalSecondaryIndexes`

Questo approccio evita la necessità di `null` controlli espliciti e si allinea alle moderne best practice Java per la gestione di strutture di dati assenti o vuote.

Esempio completo

```
package org.example;

import com.amazonaws.regions.Regions;
import com.amazonaws.services.dynamodbv2.AmazonDynamoDB;
import com.amazonaws.services.dynamodbv2.AmazonDynamoDBClientBuilder;
import com.amazonaws.services.dynamodbv2.model.DescribeTableRequest;
import com.amazonaws.services.dynamodbv2.model.DescribeTableResult;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.BillingMode;
import software.amazon.awssdk.services.dynamodb.model.DescribeTableResponse;
import software.amazon.awssdk.services.dynamodb.model.KeyType;
import software.amazon.awssdk.services.dynamodb.model.ScalarAttributeType;

import java.util.UUID;

public class DeserializationDifferences {
```

```
   private static final Logger LOGGER =
LoggerFactory.getLogger(DeserializationDifferences.class);
   private static final String TABLE_NAME = "DeserializationTable-" +
UUID.randomUUID();
   DynamoDbClient dynamoDbClientV2 = DynamoDbClient.create();
   AmazonDynamoDB dynamoDbClientV1 =
AmazonDynamoDBClientBuilder.standard().withRegion(Regions.US_EAST_1).build();

   public static void main(String[] args) {

       DeserializationDifferences difference = new DeserializationDifferences();
       difference.createTable();
       difference.deserializationDiffs();
       difference.deleteTable();
   }

   public void createTable(){
       dynamoDbClientV2.createTable(b -> b
               .billingMode(BillingMode.PAY_PER_REQUEST)
               .tableName(TABLE_NAME)
               .keySchema(b1 -> b1.attributeName("Id").keyType(KeyType.HASH))
               .attributeDefinitions(b2 ->
b2.attributeName("Id").attributeType(ScalarAttributeType.S)));
       dynamoDbClientV2.waiter().waitUntilTableExists(b -> b.tableName(TABLE_NAME));
   }

   public void deserializationDiffs(){

       DescribeTableResponse v2Response = dynamoDbClientV2.describeTable(builder ->
builder.tableName(TABLE_NAME));
       // V2 provides has* methods on model objects for list/map members. No null
check needed.
       LOGGER.info( String.valueOf(v2Response.table().hasGlobalSecondaryIndexes()) );

LOGGER.info( String.valueOf(v2Response.table().globalSecondaryIndexes().isEmpty()) );
       // V2 deserialize to an empty collection.
       LOGGER.info(v2Response.table().globalSecondaryIndexes().toString());

       // V1 deserialize to null.
       DescribeTableResult v1Result = dynamoDbClientV1.describeTable(new
DescribeTableRequest(TABLE_NAME));
       if (v1Result.getTable().getGlobalSecondaryIndexes() != null){
           LOGGER.info(v1Result.getTable().getGlobalSecondaryIndexes().toString());
       } else {
```

```
            LOGGER.info("The list of global secondary indexes returned by the V1 call
 is <null>");
        }
    }

    public void deleteTable(){
        dynamoDbClientV2.deleteTable(b -> b.tableName(TABLE_NAME));
        dynamoDbClientV2.waiter().waitUntilTableNotExists(b ->
 b.tableName(TABLE_NAME));
    }
}
```

La risposta JSON per il `describeTable` metodo del client V1 e V2 non contiene alcun attributo: `GlobalSecondaryIndexes`

```
{
  "Table": {
    "AttributeDefinitions": [
      {
        "AttributeName": "Id",
        "AttributeType": "S"
      }
    ],
    "BillingModeSummary": {
      "BillingMode": "PAY_PER_REQUEST",
      "LastUpdateToPayPerRequestDateTime": ...
    },
    "CreationDateTime": ...,
    "DeletionProtectionEnabled": false,
    "ItemCount": 0,
    "KeySchema": [
      {
        "AttributeName": "Id",
        "KeyType": "HASH"
      }
    ],
    "ProvisionedThroughput": {
      "NumberOfDecreasesToday": 0,
      "ReadCapacityUnits": 0,
      "WriteCapacityUnits": 0
    },
    "TableArn": "arn:aws:dynamodb:us-east-1:11111111111:table/
DeserializationTable-...",
```

```
    "TableId": "...",
    "TableName": "DeserializationTable-...",
    "TableSizeBytes": 0,
    "TableStatus": "ACTIVE",
    "TableThroughputModeSummary": {
      "LastUpdateToPayPerRequestDateTime": ...,
      "TableThroughputMode": "PAY_PER_REQUEST"
    },
    "WarmThroughput": {
      "ReadUnitsPerSecond": 12000,
      "Status": "ACTIVE",
      "WriteUnitsPerSecond": 4000
    }
  }
}
```

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
`com.amazonaws.SdkBaseException com.amazonaws.AmazonClientException`	`software.amazon.awssdk.core.exception.SdkException`
`com.amazonaws.SdkClientException`	`software.amazon.awssdk.core.exception.SdkClientException`
`com.amazonaws.AmazonServiceException`	`software.amazon.awssdk.awscore.exception.AwsServiceException`

La tabella seguente mappa i metodi sulle classi di eccezioni tra le versioni 1.x e 2.x.

1.x	2.x
`AmazonServiceException.getRequestId`	`SdkServiceException.requestId`
`AmazonServiceException.getServiceName`	`AwsServiceException.awsErrorDetails().serviceName`
`AmazonServiceException.getErrorCode`	`AwsServiceException.awsErrorDetails().errorCode`
`AmazonServiceException.getErrorMessage`	`AwsServiceException.awsErrorDetails().errorMessage`
`AmazonServiceException.getStatusCode`	`AwsServiceException.awsErrorDetails().sdkHttpResponse().statusCode`
`AmazonServiceException.getHttpHeaders`	`AwsServiceException.awsErrorDetails().sdkHttpResponse().headers`
`AmazonServiceException.rawResponse`	`AwsServiceException.awsErrorDetails().rawResponse`

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 per Java 2.x. Nella versione 1.x, il Amazon S3 client non viene generato direttamente dall'API del servizio. Questo provocava incoerenze tra l'API di servizio e le operazioni SDK. Nella versione 2.x, ora generiamo il Amazon S3 client in modo che sia più coerente con l'API del servizio.

La tabella seguente mostra i nomi delle operazioni nelle due versioni.

Nomi delle operazioni di Amazon S3

1.x	2.x
abortMultipartUpload	abortMultipartUpload
changeObjectStorageClass	copyObject
completeMultipartUpload	completeMultipartUpload
copyObject	copyObject
copyPart	uploadPartCopy
createBucket	createBucket
deleteBucket	deleteBucket
deleteBucketAnalyticsConfiguration	deleteBucketAnalyticsConfiguration
deleteBucketCrossOriginConfiguration	deleteBucketCors
deleteBucketEncryption	deleteBucketEncryption
deleteBucketInventoryConfiguration	deleteBucketInventoryConfiguration
deleteBucketLifecycleConfiguration	deleteBucketLifecycle
deleteBucketMetricsConfiguration	deleteBucketMetricsConfiguration
deleteBucketPolicy	deleteBucketPolicy
deleteBucketReplicationConfiguration	deleteBucketReplication
deleteBucketTaggingConfiguration	deleteBucketTagging

1.x	2.x
deleteBucketWebsiteConfiguration	deleteBucketWebsite
deleteObject	deleteObject
deleteObjectTagging	deleteObjectTagging
deleteObjects	deleteObjects
deleteVersion	deleteObject
disableRequesterPays	putBucketRequestPayment
doesBucketExist	headBucket
doesBucketExistV2	headBucket
doesObjectExist	headObject
enableRequesterPays	putBucketRequestPayment
generatePresignedUrl	S3Presigner
getBucketAccelerateConfiguration	getBucketAccelerateConfiguration
getBucketAcl	getBucketAcl
getBucketAnalyticsConfiguration	getBucketAnalyticsConfiguration
getBucketCrossOriginConfiguration	getBucketCors
getBucketEncryption	getBucketEncryption
getBucketInventoryConfiguration	getBucketInventoryConfiguration
getBucketLifecycleConfiguration	getBucketLifecycle o getBucketLifecycleConfiguration
getBucketLocation	getBucketLocation

1.x	2.x
getBucketLoggingConfiguration	getBucketLogging
getBucketMetricsConfiguration	getBucketMetricsConfiguration
getBucketNotificationConfiguration	getBucketNotification o getBucketNotificationConfiguration
getBucketPolicy	getBucketPolicy
getBucketReplicationConfiguration	getBucketReplication
getBucketTaggingConfiguration	getBucketTagging
getBucketVersioningConfiguration	getBucketVersioning
getBucketWebsiteConfiguration	getBucketWebsite
getObject	getObject
getObjectAcl	getObjectAcl
getObjectAsString	getObjectAsBytes().asUtf8String
getObjectMetadata	headObject
getObjectTagging	getObjectTagging
getResourceUrl	S3Utilities#getUrl
getS3AccountOwner	listBuckets
getUrl	S3Utilities#getUrl
headBucket	headBucket
initiateMultipartUpload	createMultipartUpload
isRequesterPaysEnabled	getBucketRequestPayment

1.x	2.x
listBucketAnalyticsConfigurations	listBucketAnalyticsConfigurations
listBucketInventoryConfigurations	listBucketInventoryConfigurations
listBucketMetricsConfigurations	listBucketMetricsConfigurations
listBuckets	listBuckets
listMultipartUploads	listMultipartUploads
listNextBatchOfObjects	listObjectsV2Paginator
listNextBatchOfVersions	listObjectVersionsPaginator
listObjects	listObjects
listObjectsV2	listObjectsV2
listParts	listParts
listVersions	listObjectVersions
putObject	putObject
restoreObject	restoreObject
restoreObjectV2	restoreObject
selectObjectContent	selectObjectContent
setBucketAccelerateConfiguration	putBucketAccelerateConfiguration
setBucketAcl	putBucketAcl
setBucketAnalyticsConfiguration	putBucketAnalyticsConfiguration

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 a cui è configurato per accedere.

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 nell'utilizzo di Amazon S3 dalla versione 1 alla versione 2 del AWS SDK per Java

AWS SDK for Java 2.x Introduce modifiche significative al client S3, tra cui una nuova struttura dei pacchetti, nomi di classe aggiornati e firme dei metodi riviste. Sebbene molti metodi possano essere migrati automaticamente dalla V1 alla V2 utilizzando lo strumento di migrazione, alcuni richiedono la migrazione manuale, come e. `listNextBatchOfObjects` `selectObjectContent` Inoltre, la V2 sostituisce alcune classi V1 come e con nuove implementazioni. `AccessControlList` `CannedAccessControlList`

Argomenti

- Differenze del client S3 tra la versione 1 e la versione 2 di AWS SDK per Java
- Modifiche in Amazon S3 Transfer Manager dalla versione 1 alla versione 2
- Modifiche nell'analisi di Amazon URIs S3 dalla versione 1 alla versione 2
- Modifiche all'API S3 Event Notifications dalla versione 1 alla versione 2

Differenze del client S3 tra la versione 1 e la versione 2 di AWS SDK per Java

In questo argomento, le differenze tra i client S3 nella versione 1 e nella versione 2 di SDK for Java sono organizzate in base al modo in cui lo strumento di migrazione può automatizzare la migrazione. Lo strumento supporta la migrazione della maggior parte dei metodi dalla V1 alla V2, ma alcuni metodi richiedono la migrazione manuale. Oltre ai metodi client S3, alcune classi S3 V1 non dispongono di un equivalente diretto alla versione V2 che richieda la migrazione manuale.

Indice

- Esempi di metodi V1 supportati dallo strumento di migrazione
  - putObject
  - getObject
- Metodi V1 che richiedono la migrazione manuale
  - La V1 getObject utilizza le versioni da V1 a V2 S3ObjectIdGetObjectRequest.builder()
  - Da V1 a V2 listNextBatchOfObjectslistObjectsV2Paginator
  - Da V1 a V2 listNextBatchOfVersionslistObjectVersionsPaginator
  - selectObjectContent
  - Il metodo da V1 setBucketAcl a V2 è attivo aclPutBucketAclRequest.builder()
  - Metodo da V1 setObjectAcl a V2 attivo aclPutObjectAclRequest.builder()
  - Da V1 a V2 initiateMultipartUploadcreateMultipartUpload
    - Esempio di migrazione
    - Differenze di implementazione
  - regionMetodo da V1 setRegion a V2 su Client Builder
  - V1 setS3ClientOptions(S3ClientOptions clientOptions)
  - Da V1 a V2 setBucketLoggingConfigurationputBucketLogging
  - Da V1 a V2 setBucketLifecycleConfigurationputBucketLifecycleConfiguration
  - Da V1 a V2 setBucketTaggingConfigurationputBucketTagging
- Classi V1 che richiedono la migrazione manuale
  - Da V1 AccessControlList a V2 AccessControlPolicy
  - Da V1 a V2 CannedAccessControlList ed enumerazioni BucketCannedACLObjectCannedACL
  - Da V1 a V2 BucketNotificationConfigurationNotificationConfiguration
  - Da V1 MultiFactorAuthentication al mfa metodo di V2 su un generatore di richieste

Esempi di metodi V1 supportati dallo strumento di migrazione

Lo strumento di migrazione migra automaticamente la maggior parte dei metodi dalla V1 alla V2. I `getObject` metodi `putObject` and sono due esempi.

`putObject`

```
// SDK V1
```

```
s3Client.putObject("my-bucket", "my-key", "Hello World!");


// SDK V2
s3Client.putObject(req -> req
    .bucket("my-bucket")
    .key("my-key"),
    RequestBody.fromString("Hello World!"));
```

`getObject`

```
// SDK V1
S3Object object = s3Client.getObject("my-bucket", "my-key");
InputStream content = object.getObjectContent();


// SDK V2
ResponseInputStream<GetObjectResponse> response = s3Client.getObject(req -> req
    .bucket("my-bucket")
    .key("my-key"));
```

Metodi V1 che richiedono la migrazione manuale

È necessario migrare manualmente i seguenti metodi client V1 S3. Quando si utilizza lo strumento di migrazione, viene visualizzato un commento nel file Java di output V2 che rimanda a questo argomento.

La V1 **`getObject`** utilizza le versioni da V1 a V2 **`S3ObjectIdGetObjectRequest.builder()`**

```
// SDK V1
AmazonS3 s3ClientV1 = AmazonS3ClientBuilder.standard()
        .withRegion(Regions.US_WEST_2)
        .build();

S3ObjectId s3ObjectId = new S3ObjectId(
    "my-bucket",
    "object-key",
    "abc123version"
);

GetObjectRequest getRequest= new GetObjectRequest(s3ObjectId);
S3Object s3ObjectVersioned = s3Client.getObject(getRequest);
```

```
// SDK V2
// V2 does not include a 'S3ObjectId' class. V2 uses the request builder pattern
// to supply the bucket, key, and version parameters.
S3Client s3Client = S3Client.builder()
    .region(Region.US_WEST_2)
    .build();

GetObjectRequest getRequest = GetObjectRequest.builder()
    .bucket("my-bucket")
    .key("object-key")
    .versionId("abc123version")
    .build();

ResponseInputStream<GetObjectResponse> response = s3Client.getObject(getRequest);
```

Da V1 a V2 **`listNextBatchOfObjectslistObjectsV2Paginator`**

```
// SDK V1
AmazonS3 s3ClientV1 = AmazonS3ClientBuilder.standard()
    .withRegion(Regions.US_WEST_2)
    .build();

ObjectListing objectListing = s3ClientV1.listObjects("bucket-name");
while (objectListing.isTruncated()) {
    objectListing = s3ClientV1.listNextBatchOfObjects(objectListing);
    for (S3ObjectSummary summary : objectListing.getObjectSummaries()) {
        System.out.println(summary.getKey());
    }
}


// SDK V2
S3Client s3ClientV2 = S3Client.builder()
    .region(Region.US_WEST_2)
    .build();

ListObjectsV2Request request = ListObjectsV2Request.builder()
    .bucket("bucket-name")
    .build();

// V2 returns a paginator.
```

```
ListObjectsV2Iterable responses = s3Client.listObjectsV2Paginator(request);

for (ListObjectsV2Response page : responses) {
    page.contents().forEach(content -> {
        System.out.println(content.key());
    });
}
```

Da V1 a V2 **listNextBatchOfVersionslistObjectVersionsPaginator**

```
// SDK V1
AmazonS3 s3ClientV1 = AmazonS3ClientBuilder.standard()
    .withRegion(Regions.US_WEST_2)
    .build();

VersionListing versionListing = s3ClientV1.listVersions("bucket-name", "prefix");
while (versionListing.isTruncated()) {
    versionListing = s3ClientV1.listNextBatchOfVersions(versionListing);
    for (S3VersionSummary version : versionListing.getVersionSummaries()) {
        System.out.println(version.getKey() + " " + version.getVersionId());
    }
}


// SDK V2
S3Client s3ClientV2 = S3Client.builder()
    .region(Region.US_WEST_2)
    .build();

ListObjectVersionsRequest request = ListObjectVersionsRequest.builder()
    .bucket("bucket-name")
    .prefix("prefix")
    .build();

// V2 returns a paginator.
ListObjectVersionsIterable responses = s3ClientV2.listObjectVersionsPaginator(request);

for (ListObjectVersionsResponse page : responses) {
    page.versions().forEach(version -> {
        System.out.println(version.key() + " " + version.versionId());
    });
}
```

`selectObjectContent`

```
// SDK V1
AmazonS3 s3ClientV1 = AmazonS3ClientBuilder.standard()
    .withRegion(Regions.US_WEST_2)
    .build();

SelectObjectContentRequest request = new SelectObjectContentRequest()
    .withBucket("bucket-name")
    .withKey("object-key")
    .withExpression("select * from S3Object")
    .withExpressionType(ExpressionType.SQL)


SelectObjectContentResult result = s3ClientV1.selectObjectContent(request);
InputStream resultInputStream = result.getPayload().getRecordsInputStream();


// SDK V2
// In V2, 'selectObjectContent()' is available only on the S3AsyncClient.
// V2 handles responses using an event-based 'SelectObjectContentEventStream'.
S3AsyncClient s3ClientV2 = S3AsyncClient.builder()
    .region(Region.US_WEST_2)
    .build();

SelectObjectContentRequest request = SelectObjectContentRequest.builder()
    .bucket("bucket-name")
    .key("object-key")
    .expression("select * from S3Object")
    .expressionType(ExpressionType.SQL)
    .build();

SelectObjectContentResponseHandler handler = new SelectObjectContentResponseHandler() {
    // Implement the required abstract methods such as 'onEventStream()'.
};

CompletableFuture<Void> future = s3ClientV2.selectObjectContent(request, handler);
// The 'SelectObjectContentResponseHandler' implementation processes the results.
```

Il metodo da V1 **`setBucketAcl`** a V2 è attivo **`aclPutBucketAclRequest.builder()`**

```
// SDK V1
AmazonS3 s3ClientV1 = AmazonS3ClientBuilder.standard()
```

```
    .withRegion(Regions.US_WEST_2)
    .build();

AccessControlList acl = new AccessControlList();
acl.grantPermission(GroupGrantee.AllUsers, Permission.Read);
s3ClientV1.setBucketAcl("bucket-name", acl);


// SDK V2
S3Client s3ClientV2 = S3Client.builder()
    .region(Region.US_WEST_2)
    .build();

PutBucketAclRequest request = PutBucketAclRequest.builder()
    .bucket("bucket-name")
    .acl(BucketCannedACL.PRIVATE)
    .build();

s3ClientV2.putBucketAcl(request);
```

Metodo da V1 **`setObjectAcl`** a V2 attivo **`aclPutObjectAclRequest.builder()`**

```
// SDK V1
AmazonS3 s3ClientV1 = AmazonS3ClientBuilder.standard()
    .withRegion(Regions.US_WEST_2)
    .build();

AccessControlList acl = new AccessControlList();
acl.grantPermission(GroupGrantee.AllUsers, Permission.Read);
s3ClientV1.setObjectAcl("bucket-name", "object-key", acl);


// SDK V2
S3Client s3ClientV2 = S3Client.builder()
    .region(Region.US_WEST_2)
    .build();

PutObjectAclRequest request = PutObjectAclRequest.builder()
    .bucket("bucket-name")
    .key("object-key")
    .acl(ObjectCannedACL.PRIVATE)
    .build();
```

```
s3ClientV2.putObjectAcl(request);
```

Da V1 a V2 **initiateMultipartUploadcreateMultipartUpload**

Esempio di migrazione

```
// SDK V1
AmazonS3 s3ClientV1 = AmazonS3ClientBuilder.standard()
    .withRegion(Regions.US_WEST_2)
    .build();

ObjectMetadata metadata = new ObjectMetadata();
metadata.setContentType("application/zip");
metadata.addUserMetadata("mykey", "myvalue");

InitiateMultipartUploadRequest initRequest = new InitiateMultipartUploadRequest(
    "bucket-name",
    "object-key",
    metadata
);

InitiateMultipartUploadResult initResponse =
 s3ClientV1.initiateMultipartUpload(initRequest);
String uploadId = initResponse.getUploadId();


// SDK V2
// V1 uses ObjectMetadata methods, whereas V2 uses a simple Map.
S3Client s3ClientV2 = S3Client.builder()
    .region(Region.US_WEST_2)
    .build();

CreateMultipartUploadRequest createMultipartRequest =
 CreateMultipartUploadRequest.builder()
    .bucket("my-bucket")
    .key("object-key")
    .contentType("application/zip")
    .metadata(Collections.singletonMap("mykey", "myvalue"))
    .build();

CreateMultipartUploadResponse response =
 s3ClientV2.createMultipartUpload(createMultipartRequest);
String uploadId = response.uploadId();
```

Differenze di implementazione

Il valore di `Content-Type` intestazione predefinito per i seguenti metodi è diverso come illustrato nella tabella seguente.

Versione SDK	Metodo	Valore predefinito **`Content-Type`**
versione 1	`initiateMultipartUpload`	`application/octet-stream`
versione 2	`createMultipartUpload`	`binary/octet-stream`

`region`Metodo da V1 **`setRegion`** a V2 su Client Builder

```
// SDK V1
AmazonS3 s3ClientV1 = AmazonS3ClientBuilder.standard().build();
s3ClientV1.setRegion(Region.getRegion(Regions.US_WEST_2));


// SDK V2
S3Client s3ClientV2 = S3Client.builder()
    .region(Region.US_WEST_2)
    .build();
```

V1 **`setS3ClientOptions(S3ClientOptions clientOptions)`**

Invece di utilizzare un singolo `S3ClientOptions` oggetto con il `setS3ClientOptions` metodo, V2 fornisce metodi sul client builder per impostare le opzioni.

Da V1 a V2 **`setBucketLoggingConfigurationputBucketLogging`**

```
// SDK V1
AmazonS3 s3ClientV1 = AmazonS3ClientBuilder.standard()
    .withRegion(Regions.US_WEST_2)
    .build();

BucketLoggingConfiguration loggingConfig = new BucketLoggingConfiguration();
loggingConfig.setDestinationBucketName("log-bucket");
```

```
SetBucketLoggingConfigurationRequest request = new
 SetBucketLoggingConfigurationRequest(
    "source-bucket",
    loggingConfig
);

s3ClientV1.setBucketLoggingConfiguration(request);


// SDK V2
// In V2, V1's 'BucketLoggingConfiguration' is replaced by 'BucketLoggingStatus'
// and 'LoggingEnabled'.
S3Client s3ClientV2 = S3Client.builder()
    .region(Region.US_WEST_2)
    .build();

LoggingEnabled loggingEnabled = LoggingEnabled.builder()
    .targetBucket("log-bucket")
    .build();

BucketLoggingStatus loggingStatus = BucketLoggingStatus.builder()
    .loggingEnabled(loggingEnabled)
    .build();

PutBucketLoggingRequest request = PutBucketLoggingRequest.builder()
    .bucket("source-bucket")
    .bucketLoggingStatus(loggingStatus)
    .build();

s3ClientV2.putBucketLogging(request);
```

Da V1 a V2 **setBucketLifecycleConfigurationputBucketLifecycleConfiguration**

```
// SDK V1
BucketLifecycleConfiguration.Rule rule = new BucketLifecycleConfiguration.Rule()
    .withId("Archive and Delete Rule")
    .withPrefix("documents/")
    .withStatus(BucketLifecycleConfiguration.ENABLED)
    .withTransitions(Arrays.asList(
        new Transition()
            .withDays(30)
            .withStorageClass(StorageClass.StandardInfrequentAccess.toString()),
```

```
        new Transition()
            .withDays(90)
            .withStorageClass(StorageClass.Glacier.toString())
    ))
    .withExpirationInDays(365);

BucketLifecycleConfiguration configuration = new BucketLifecycleConfiguration()
    .withRules(Arrays.asList(rule));

s3ClientV1.setBucketLifecycleConfiguration("my-bucket", configuration);


// SDK V2
LifecycleRule rule = LifecycleRule.builder()
    .id("Archive and Delete Rule")
    .filter(LifecycleRuleFilter.builder()
        .prefix("documents/")
        .build())
    .status(ExpirationStatus.ENABLED)
    .transitions(Arrays.asList(
        Transition.builder()
            .days(30)
            .storageClass(TransitionStorageClass.STANDARD_IA)
            .build(),
        Transition.builder()
            .days(90)
            .storageClass(TransitionStorageClass.GLACIER)
            .build()
    ))
    .expiration(LifecycleExpiration.builder()
        .days(365)
        .build())
    .build();

PutBucketLifecycleConfigurationRequest request =
 PutBucketLifecycleConfigurationRequest.builder()
    .bucket("my-bucket")
    .lifecycleConfiguration(BucketLifecycleConfiguration.builder()
        .rules(rule)
        .build())
    .build();

s3ClientV2.putBucketLifecycleConfiguration(request);
```

Da V1 a V2 **setBucketTaggingConfigurationputBucketTagging**

```
// SDK V1
List<TagSet> tagsets = new ArrayList<>();
TagSet tagSet = new TagSet();
tagSet.setTag("key1", "value1");
tagSet.setTag("key2", "value2");
tagsets.add(tagSet);

BucketTaggingConfiguration bucketTaggingConfiguration = new
 BucketTaggingConfiguration();
bucketTaggingConfiguration.setTagSets(tagsets);

SetBucketTaggingConfigurationRequest request = new
 SetBucketTaggingConfigurationRequest(
    "my-bucket",
    bucketTaggingConfiguration
);

s3ClientV1.setBucketTaggingConfiguration(request);


// SDK V2
Tagging tagging = Tagging.builder()
    .tagSet(Arrays.asList(
        Tag.builder()
            .key("key1")
            .value("value1")
            .build(),
        Tag.builder()
            .key("key2")
            .value("value2")
            .build()
    ))
    .build();

PutBucketTaggingRequest request = PutBucketTaggingRequest.builder()
    .bucket("my-bucket")
    .tagging(tagging)
    .build();

s3ClientV2.putBucketTagging(request);
```

Classi V1 che richiedono la migrazione manuale

Da V1 **`AccessControlList`** a V2 **`AccessControlPolicy`**

```
// SDK V1
AccessControlList aclV1 = new AccessControlList();
aclV1.setOwner(new Owner("owner-id", "owner-name"));
aclV1.grantPermission(GroupGrantee.AllUsers, Permission.Read);

// SDK V2
// To migrate from V1 to V2, replace direct 'AccessControlList' modifications with an
// 'AccessControlPolicy.builder()' that contains both owner information and grants.
// Note that V2's approach requires building the complete permission set upfront rather
 than modifying permissions incrementally.
AccessControlPolicy acpV2 = AccessControlPolicy.builder()
    .owner(Owner.builder()
        .id("owner-id")
        .displayName("owner-name")
        .build())
    .grants(Arrays.asList(
         Grant.builder()
            .grantee(Grantee.builder()
                 .type(Type.GROUP)
                 .uri("http://acs.amazonaws.com/groups/global/AllUsers")
                 .build())
             .permission(Permission.READ)
             .build()
     ))
     .build();
```

Da V1 a V2 **`CannedAccessControlList`** ed enumerazioni **`BucketCannedACL`** **`ObjectCannedACL`**

```
// SDK V1
// In V1, 'CannedAccessControlList' is an enumeration of predefined ACLs.
AmazonS3 s3ClientV1 = AmazonS3ClientBuilder.standard().build();

// Creating a bucket.
s3ClientV1.setBucketAcl("bucket-name", CannedAccessControlList.PublicRead);

// Creating an object.
PutObjectRequest putObjectRequest = new PutObjectRequest("bucket-name", "object-key",
 file)
```

```
    .withCannedAcl(CannedAccessControlList.PublicRead);
s3ClientV1.putObject(putObjectRequest);


// SDK V2
// V2 replaces V1's 'CannedAccessControlList' with 'BucketCannedACL' for buckets and
 'ObjectCannedACL' for objects.
S3Client s3ClientV2 = S3Client.builder().build();

// Creating a bucket.
PutBucketAclRequest bucketRequest = PutBucketAclRequest.builder()
    .bucket("bucket-name")
    .acl(BucketCannedACL.PRIVATE)
    .build();
s3ClientV2.putBucketAcl(bucketRequest);

// Creating an object.
PutObjectRequest objectRequest = PutObjectRequest.builder()
    .bucket("bucket-name")
    .key("object-key")
    .acl(ObjectCannedACL.PUBLIC_READ)
    .build();
s3ClientV2.putObject(objectRequest, RequestBody.fromFile(file));
```

Da V1 a V2 **BucketNotificationConfigurationNotificationConfiguration**

```
//SDK V1
BucketNotificationConfiguration notificationConfig = new
 BucketNotificationConfiguration();

// Adding configurations by name
notificationConfig.addConfiguration("lambdaConfig",
    new LambdaConfiguration("arn:aws:lambda:function", "s3:ObjectCreated:"));

notificationConfig.addConfiguration("topicConfig",
    new TopicConfiguration("arn:aws:sns:topic", "s3:ObjectRemoved:"));

notificationConfig.addConfiguration("queueConfig",
    new QueueConfiguration("arn:aws:sqs:queue", "s3:ObjectRestore:*"));

s3Client.setBucketNotificationConfiguration("bucket", notificationConfig);
```

```
// SDK V2
// In V2, V1's BucketNotificationConfiguration is renamed to
 NotificationConfiguration.
// V2 contains no common abstract class for LambdaFunction/Topic/Queue configurations.
NotificationConfiguration notificationConfig = NotificationConfiguration.builder()
    .lambdaFunctionConfigurations(
        LambdaFunctionConfiguration.builder()
            .lambdaFunctionArn("arn:aws:lambda:function")
            .events(Event.valueOf("s3:ObjectCreated:"))
            .build())
    .topicConfigurations(
        TopicConfiguration.builder()
            .topicArn("arn:aws:sns:topic")
            .events(Event.valueOf("s3:ObjectRemoved:"))
            .build())
    .queueConfigurations(
        QueueConfiguration.builder()
            .queueArn("arn:aws:sqs:queue")
            .events(Event.valueOf("s3:ObjectRestore:*"))
            .build())
    .build();

s3Client.putBucketNotificationConfiguration(req -> req
    .bucket("bucket")
    .notificationConfiguration(notificationConfig));
```

Da V1 **`MultiFactorAuthentication`** al **`mfa`** metodo di V2 su un generatore di richieste

```
// SDK V1
AmazonS3 s3ClientV1 = AmazonS3ClientBuilder.standard()
    .withRegion(Regions.US_WEST_2)
    .build();

BucketVersioningConfiguration versioningConfig = new BucketVersioningConfiguration()
    .withStatus(BucketVersioningConfiguration.ENABLED);

// Create an MFA configuration object.
MultiFactorAuthentication mfa = new MultiFactorAuthentication(
    "arn:aws:iam::1234567890:mfa/user",
    "123456"
);

// Create the request object.
```

```
SetBucketVersioningConfigurationRequest request = new
 SetBucketVersioningConfigurationRequest(
    "bucket-name",
    versioningConfig,
    mfa
);

// Send the request.
s3ClientV1.setBucketVersioningConfiguration(request);


// SDK V2
// V2 replaces V1's MultiFactorAuthentication POJO with parameters you set on the
 request builder.
S3Client s3ClientV2 = S3Client.builder()
    .region(Region.US_WEST_2)
    .build();

PutBucketVersioningRequest request = PutBucketVersioningRequest.builder()
    .bucket("bucket-name")
    .versioningConfiguration(VersioningConfiguration.builder()
        .status(BucketVersioningStatus.ENABLED)
        .build())
    .mfa("arn:aws:iam::1234567890:mfa/user 123456")  // Single method takes both MFA
 erial number and token.
    .build();

s3ClientV2.putBucketVersioning(request);
```

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

Modifica	v1	v2
	<groupId>com.amazonaws</groupId> <artifactId>aws-java-sdk-bom</artifactId> <version>*1.12.691*[1]</version> <type>pom</type> <scope>import</scope> </dependency> </dependencies> </dependencyManagement> <dependencies> <dependency> <groupId>com.amazonaws</groupId> <artifactId>aws-java-sdk-s3</artifactId> </dependency> </dependencies>	<groupId>software.amazon.awssdk</groupId> <artifactId>bom</artifactId> <version>*2.27.21*[2]</version> <type>pom</type> <scope>import</scope> </dependency> </dependencies> </dependencyManagement> <dependencies> <dependency> <groupId>software.amazon.awssdk</groupId> <artifactId>s3-transfer-manager</artifactId> </dependency> **// Add the following if using the** **// AWS CRT-based S3 client.** <dependency> <groupId>software.amazon.awssdk.crt</groupId> <artifactId>aws-crt</artifactId> <version>*0.29.14*[3]</version> </dependency> </dependencies>

Modifica	v1	v2
Nome pacchetto	`com.amazonaws.services.s3.transfer`	`software.amazon.awssdk.transfer.s3`
Class name (Nome classe)	`TransferManager`	`S3TransferManager`

[1] Ultima versione. [2] Ultima versione. [3] Ultima versione.

Modifiche di configurazione

Le modifiche alla configurazione che devi impostare per il gestore di trasferimento v2 dipendono dal client S3 che utilizzi. Puoi scegliere tra il client S3 basato su AWS CRT o il client asincrono S3 standard basato su Java. Per informazioni sulle differenze, consulta l'argomento. the section called "Client S3 nell'SDK"

Use the AWS CRT-based S3 client

Impostazione	v1	v2 - Transfer Manager che utilizza un client S3 basato su AWS CRT
(procurati un costruttore)	`TransferManagerBuilder tmBuilder = TransferManagerBuilder.standard();`	`S3TransferManager.Builder tmBuilder = S3TransferManager.builder();`
Client S3	`tmBuilder.withS3Client(...); tmBuilder.setS3Client(...);`	`tmBuilder.s3Client(...);`
Esecutore	`tmBuilder.withExecutorFactory(...); tmBuilder.setExecutorFactory(...);`	`tmBuilder.executor(...);`

Impostazione	v1	v2 - Transfer Manager che utilizza un client S3 basato su AWS CRT
Chiudi i pool di thread	`tmBuilder.withShutDownThreadPools(...);` `tmBuilder.setShutdownThreadPools(...);`	Non supportato. L'esecutore fornito non verrà spento quando viene chiuso `S3TransferManager`

Dimensione minima della parte da caricare	`tmBuilder.withMinimumUploadPartSize(...);`
	`tmBuilder.setMinimumUploadPartSize(...);`
		`S3AsyncClient s3 = S3AsyncClient.crtBuilder().minimumPartSizeInBytes(...).build();`
		
		`tmBuilder.s3Client(s3);`

Soglia di caricamento in più parti	`tmBuilder.withMinimumUploadPartSize(...);`
	`tmBuilder.setMinimumUploadPartSize(...);`
		`S3AsyncClient s3 = S3AsyncClient.crtBuilder().thresholdInBytes(...).build();`
		
		`tmBuilder.s3Client(s3);`

Impostazione	v1	v2 - Transfer Manager che utilizza un client S3 basato su AWS CRT

Dimensione minima della parte di copia	`tmBuilder.withMinimumUploadPartSize(...);`
	`tmBuilder.setMinimumUploadPartSize(...);`
		`S3AsyncClient s3 = S3AsyncClient.crtBuilder().minimumPartSizeInBytes(...).build();`
		
		`tmBuilder.s3Client(s3);`

Soglia di copia in più parti	`tmBuilder.withMinimumUploadPartSize(...);`
	`tmBuilder.setMinimumUploadPartSize(...);`
		`S3AsyncClient s3 = S3AsyncClient.crtBuilder().thresholdInBytes(...).build();`
		
		`tmBuilder.s3Client(s3);`

Disattiva i download paralleli	`tmBuilder.withDisableParallelDownloads(...);`
	`tmBuilder.setDisableParallelDownloads(...);`
		Disabilita i download paralleli passando un client S3 standard basato su Java con multipart disabilitato (impostazione predefinita) al gestore di trasferimento.
		
		`S3AsyncClient s3 = S3AsyncClient.builder().build();`
		
		`tmBuilder.s3Client(s3);`

Impostazione	v1	v2 - Transfer Manager che utilizza un client S3 basato su AWS CRT
Calcola sempre md5 multipart	`tmBuilder.withAlwaysCalculateMultipartMd5(...);` `tmBuilder.setAlwaysCalculateMultipartMd5(...);`	Non supportato.

Use Java-based S3 async client

Impostazione	v1	v2 - Transfer Manager che utilizza un client asincrono S3 basato su Java
(procurati un costruttore)	`TransferManagerBuilder tmBuilder = TransferManagerBuilder.standard();`	`S3TransferManager.Builder tmBuilder = S3TransferManager.builder();`
Client S3	`tmBuilder.withS3Client(...);` `tmBuilder.setS3Client(...);`	`tmBuilder.s3Client(...);`
Esecutore	`tmBuilder.withExecutorFactory(...);` `tmBuilder.setExecutorFactory(...);`	`tmBuilder.executor(...);`
Chiudi i pool di thread	`tmBuilder.withShutDownThreadPools(...);`	Non supportato. L'esecutore fornito non verrà spento quando viene chiuso `S3TransferManager`

Impostazione	v1	v2 - Transfer Manager che utilizza un client asincrono S3 basato su Java
	`tmBuilder.setShutdownThreadPools(...);`	

Dimensione minima della parte da caricare	`tmBuilder.withMinimumUploadPartSize(...);`
	`tmBuilder.setMinimumUploadPartSize(...);`
		`S3AsyncClient s3 = S3AsyncClient.builder()`
		`.multipartConfiguration(cfg ->`
		`cfg.minimumPartSizeInBytes(...)).build();`
		
		`tmBuilder.s3Client(s3);`

Soglia di caricamento in più parti	`tmBuilder.withMinimumUploadPartSize(...);`
	`tmBuilder.setMinimumUploadPartSize(...);`
		`S3AsyncClient s3 = S3AsyncClient.builder()`
		`.multipartConfiguration(cfg ->`
		`cfg.thresholdInBytes(...)).build();`
		
		`tmBuilder.s3Client(s3);`

Impostazione	v1	v2 - Transfer Manager che utilizza un client asincrono S3 basato su Java

Dimensione minima della parte di copia	`tmBuilder.withMinimumUploadPartSize(...);`
	`tmBuilder.setMinimumUploadPartSize(...);`
		`S3AsyncClient s3 = S3AsyncClient.builder()`
		`.multipartConfiguration(cfg ->`
		`cfg.minimumPartSizeInBytes(...)).build();`
		
		`tmBuilder.s3Client(s3);`

Soglia di copia in più parti	`tmBuilder.withMinimumUploadPartSize(...);`
	`tmBuilder.setMinimumUploadPartSize(...);`
		`S3AsyncClient s3 = S3AsyncClient.builder()`
		`.multipartConfiguration(cfg ->`
		`cfg.thresholdInBytes(...)).build();`
		
		`tmBuilder.s3Client(s3);`

Impostazione	v1	v2 - Transfer Manager che utilizza un client asincrono S3 basato su Java

Disattiva i download paralleli	`tmBuilder.withDisableParallelDownloads(...);`
	`tmBuilder.setDisableParallelDownloads(...);`
		Disabilita i download paralleli passando un client S3 standard basato su Java con multipart disabilitato (impostazione predefinita) al gestore di trasferimento.
		`S3AsyncClient s3 = S3AsyncClient.builder().build();`
		`tmBuilder.s3Client(s3);`

Calcola sempre md5 multipart	`tmBuilder.withAlwaysCalculateMultipartMd5(...);` `tmBuilder.setAlwaysCalculateMultipartMd5(...);`	Non supportato.

Modifiche al comportamento

Requisiti di trasferimento parallelo

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 e AWS il client asincrono S3 basato su Java. Per utilizzare il client S3 AWS basato su CRT, è necessario aggiungere esplicitamente la dipendenza dalla libreria Common Runtime (CRT) per massimizzare le prestazioni.AWS Per utilizzare il client asincrono S3 basato su Java con multipart abilitato, è necessario utilizzare l'SDK alla versione o superiore. *`2.25.X <TODO>`*

Il solo client S3 AWS basato su CRT, senza utilizzo, offre prestazioni massimizzate per i trasferimenti `S3TransferManager` paralleli. `S3TransferManagerv2` fornisce funzionalità aggiuntive APIs che semplificano il trasferimento di file e directory.

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'analisi di Amazon URIs S3 dalla versione 1 alla versione 2

Questo argomento descrive in dettaglio le modifiche apportate all'analisi di Amazon URIs 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
	`<dependencyManagement>`	`<dependencyManagement>`

Modifica	v1	v2
Dipendenze da Maven	<dependencies> <dependency> <groupId> com.amazonaws</gro upId> <artifact Id>aws-java-sdk-bom</ artifactId> <version> *1.12.587*[1]</version> <type>pom</ type> <scope>im port</scope> </dependency> </dependencies> </dependencyManageme nt> <dependencies> <dependency> <groupId> com.amazonaws</gro upId> <artifact Id>s3</artifactId> </dependency> </dependencies>	<dependencies> <dependency> <groupId> software.amazon.aw ssdk</groupId> <artifact Id>bom</artifactId> <version> *2.27.21*[2]</version> <type>pom</ type> <scope>im port</scope> </dependency> </dependencies> </dependencyManageme nt> <dependencies> <dependency> <groupId> software.amazon.aw ssdk</groupId> <artifact Id>s3</artifactId> </dependency> </dependencies>
Nome pacchetto	`com.amazonaws.serv ices.s3`	`software.amazon.aw ssdk.services.s3`
Nomi delle classi	`AmazonS3URI`	`S3URI`

[1] Ultima versione. [2] Ultima versione.

Modifiche alle API

Comportamento	v1	v2

Analizza un URI S3.	`URI uri = URI.creat e( "https://s3.amazon aws.com");`
	
	`AmazonS3Uri s3Uri = new AmazonS3U RI(uri, false);`
		`S3Client s3Client = S3Client.create(); S3Utilities s3Utiliti es = s3Client.utilities ();`
		
		`S3Uri s3Uri = s3Utilities.parseU ri(uri);`

Recupera il nome del bucket da un URI S3.	`String bucket = s3Uri.getBucket();`	`Optional<String> bucket = s3Uri.bucket();`
Recupera la chiave.	`String key = s3Uri.get Key();`	`Optional<String> key = s3Uri.key();`

Recupera la regione.	`String region = s3Uri.getRegion();`	`Optional<Region> region = s3Uri.region();`
		
		`String region; if (s3Uri.region().is Present()) { region = s3Uri.reg ion().get().id(); }`

Recupera se l'URI S3 è in stile percorso.	`boolean isPathStyle = s3Uri.isPathStyle();`	`boolean isPathStyle = s3Uri.isPathStyle();`
Recupera l'ID della versione.	`String versionId = s3Uri.getVersionId();`	`Optional<String> versionId =`

Comportamento	v1	v2
		`s3Uri.firstMatchingRawQueryParameter("versionId");`
Recupera i parametri dell'interrogazione.	N/D	`Map<String, List<String>> queryParams = s3Uri.rawQueryParameters();`

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 all'API S3 Event Notifications dalla versione 1 alla versione 2

Questo argomento descrive in dettaglio le modifiche all'API S3 Event Notifications dalla versione 1.x (v1) alla versione 2 .x (v2) di. AWS SDK per Java

Modifiche di alto livello

Modifiche strutturali

La V1 utilizza classi interne statiche per i `EventNotificationRecord` tipi e i relativi attributi, mentre la v2 utilizza classi pubbliche separate per `EventNotificationRecord` i tipi.

Modifiche alla convenzione di denominazione

In v1, i nomi delle classi di attributi includono il suffisso Entity, mentre v2 omette questo suffisso per una denominazione più semplice: ad esempio, EventData anziché. eventDataEntity

Modifiche nelle dipendenze, nei pacchetti e nei nomi delle classi

Nella v1, le classi API di notifica degli eventi S3 vengono importate in modo transitivo insieme al modulo S3 (artifactID). `aws-java-sdk-s3` Tuttavia, nella v2, è necessario aggiungere una dipendenza dall'artefatto. `s3-event-notifications`

Modifica	v1	v2
Dipendenze da Maven	<dependencyManagement> <dependencies> <dependency> <groupId> com.amazonaws</gro upId> <artifact Id>aws-java-sdk-bom</ artifactId> <version> *1.X.X*</version> <type>pom</ type> <scope>im port</scope> </dependency> </dependencies> </dependencyManageme nt> <dependencies> <dependency> <groupId> com.amazonaws</gro upId> <artifact Id>aws-java-sdk-s3</ artifactId> </dependency> </dependencies>	<dependencyManagement> <dependencies> <dependency> <groupId> software.amazon.aw ssdk</groupId> <artifact Id>bom</artifactId> <version> *2.X.X*[1]</version> <type>pom</ type> <scope>im port</scope> </dependency> </dependencies> </dependencyManageme nt> <dependencies> <dependency> <groupId> software.amazon.aw ssdk</groupId> <artifactId>s3- event-notifications</ artifactId> </dependency> </dependencies>
Nome pacchetto	`com.amazonaws.serv ices.s3.event`	`software.amazon.aw ssdk.eventnotifica tions.s3.model`

Modifica	v1	v2
Nomi delle classi	S3EventNotification S3 S.3 EventNotification EventNotificationRecord S3EventNotification. GlacierEventDataEntity S3EventNotification. IntelligentTieringEventDataEntity S3EventNotification. LifecycleEventDataEntity S3EventNotification. ReplicationEventDataEntity S3EventNotification. RequestParametersEntity S3EventNotification. ResponseElementsEntity S3EventNotification. RestoreEventDataEntity S3 S3 EventNotification BucketEntity Entità S3 EventNotification .S3 S3 EventNotification S.3 ObjectEntity S3EventNotification. TransitionEventDataEntity	S3EventNotification S3EventNotificationRecord GlacierEventData IntelligentTieringEventData LifecycleEventData ReplicationEventData RequestParameters ResponseElements RestoreEventData S3Bucket S3 Oggetto S3 TransitionEventData UserIdentity

Modifica	v1	v2
	S3EventNotification. UserIdentityEntity	

[1] Ultima versione.

Modifiche alle API

JSON verso **`S3EventNotification`** e viceversa

Caso d'uso	v1	v2
Crea `S3EventNotification` da JSON String	`S3EventNotification notification = S3EventNotification.parseJson(message.body());`	`S3EventNotification notification = S3EventNotification.fromJson(message.body());`
Converti in `S3EventNotification` stringa JSON	`String json = notification.toJson();`	`String json = notification.toJson();`

Accedi agli attributi di **`S3EventNotification`**

Caso d'uso	v1	v2
Recuperare i record da una notifica	`List<S3EventNotification.S3EventNotificationRecord> records = notifcation.getRecords();`	`List<S3EventNotificationRecord> records = notification.getRecords();`
Recuperare un record da un elenco di record	`S3EventNotification.S3EventNotificationRecord record =`	`S3EventNotificationRecord record =`

Caso d'uso	v1	v2
	`records.stream().findAny().get();`	`records.stream().findAny().get();`
Recupera i dati degli eventi Glacier	`S3EventNotification.GlacierEventDataEntity glacierEventData = record.getGlacierEventData();`	`GlacierEventData glacierEventData = record.getGlacierEventData();`
Recupera i dati degli eventi di ripristino da un evento Glacier	`S3EventNotification.RestoreEventDataEntity restoreEventData = glacierEventData.getRestoreEventDataEntity();`	`RestoreEventData restoreEventData = glacierEventData.getRestoreEventData();`
Recupera i parametri della richiesta	`S3EventNotification.RequestParametersEntity requestParameters = record.getRequestParameters();`	`RequestParameters requestParameters = record.getRequestParameters();`
Recupera i dati degli eventi di Intelligent Tiering	`S3EventNotification.IntelligentTieringEventDataEntity tieringEventData = record.getIntelligentTieringEventData();`	`IntelligentTieringEventData intelligentTieringEventData = record.getIntelligentTieringEventData();`
Recupera i dati degli eventi del ciclo di vita	`S3EventNotification.LifecycleEventDataEntity lifecycleEventData = record.getLifecycleEventData();`	`LifecycleEventData lifecycleEventData = record.getLifecycleEventData();`

Caso d'uso	v1	v2
Recupera il nome dell'evento come enum	`S3Event eventNameAsEnum = record.getEventNameAsEnum();`	`//getEventNameAsEnum does not exist; use 'getEventName()' String eventName = record.getEventName();`
Recupera i dati degli eventi di replica	`S3EventNotification.ReplicationEventDataEntity replicationEntity = record.getReplicationEventDataEntity();`	`ReplicationEventData replicationEventData = record.getReplicationEventData();`
Recupera le informazioni sul bucket e sull'oggetto S3	`S3EventNotification.S3Entity s3 = record.getS3();`	`S3 s3 = record.getS3();`
Recupera le informazioni sull'identità dell'utente	`S3EventNotification.UserIdentityEntity userIdentity = record.getUserIdentity();`	`UserIdentity userIdentity = record.getUserIdentity();`
Recupera gli elementi di risposta	`S3EventNotification.ResponseElementsEntity responseElements = record.getResponseElements();`	`ResponseElements responseElements = record.getResponseElements();`

Migra la versione della libreria **`aws-lambda-java-events`**

Se utilizzi eventi aws-lambda-java-eventsdi notifica S3 all'interno di una funzione Lambda, ti consigliamo di eseguire l'aggiornamento alla versione 3.x.x più recente. Le versioni recenti eliminano tutte le dipendenze da AWS SDK per Java 1.x dall'API di notifica degli eventi di S3.

Per ulteriori informazioni sulle differenze nella gestione delle notifiche degli eventi S3 tra la `aws-lambda-java-events` libreria e l'SDK for Java 2.x, consulta. the section called “Opzioni di gestione delle notifiche degli eventi S3”

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 and. `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_ACCES S_KEY_ID AWS_ACCES S_KEY	aws.accessKeyId	AWS_ACCES S_KEY_ID	aws.accessKeyId
AWS_SECRE T_KEY AWS_SECRE T_ACCESS_KEY	aws.secretKey	AWS_SECRE T_ACCESS_KEY	aws.secre tAccessKey
AWS_SESSI ON_TOKEN	aws.sessi onToken	AWS_SESSI ON_TOKEN	aws.sessi onToken
AWS_REGION	aws.region	AWS_REGION	aws.region
AWS_CONFIG_FILE		AWS_CONFIG_FILE	aws.configFile
AWS_CREDE NTIAL_PRO FILES_FILE		AWS_SHARE D_CREDENT IALS_FILE	aws.share dCredenti alsFile
AWS_PROFILE	aws.profile	AWS_PROFILE	aws.profile
AWS_EC2_M ETADATA_D ISABLED	com.amazo naws.sdk.	AWS_EC2_M ETADATA_D ISABLED	aws.disab leEc2Metadata

1.x Variabile di ambiente	1.x Proprietà del sistema	Variabile d'ambiente 2.x	Proprietà di sistema 2.x
	disableEc2Metadata		
	com.amazonaws.sdk.ec2MetadataServiceEndpointOverride	AWS_EC2_METADATA_SERVICE_ENDPOINT	aws.ec2MetadataServiceEndpoint
AWS_CONTAINER_CREDENTIALS_RELATIVE_URI		AWS_CONTAINER_CREDENTIALS_RELATIVE_URI	aws.containerCredentialsPath
AWS_CONTAINER_CREDENTIALS_FULL_URI		AWS_CONTAINER_CREDENTIALS_FULL_URI	aws.containerCredentialsFullUri
AWS_CONTAINER_AUTHORIZATION_TOKEN		AWS_CONTAINER_AUTHORIZATION_TOKEN	aws.containerAuthorizationToken
AWS_CBOR_DISABLED	com.amazonaws.sdk.disableCbor	CBOR_ENABLED	aws.cborEnabled
AWS_ION_BINARY_DISABLE	com.amazonaws.sdk.disableIonBinary	BINARY_ION_ENABLED	aws.binaryIonEnabled

1.x Variabile di ambiente	1.x Proprietà del sistema	Variabile d'ambiente 2.x	Proprietà di sistema 2.x
AWS_EXECUTION_ENV		AWS_EXECUTION_ENV	aws.executionEnvironment
	com.amazonaws.sdk.disableCertChecking	Non supportato (funzionalità di richiesta)	Non supportato (funzionalità di richiesta)
	com.amazonaws.sdk.enableDefaultMetrics	Non supportata	Non supportato
	com.amazonaws.sdk.enableThrottledRetry	Non supportato	Non supportato
	com.amazonaws.regions.RegionUtils.fileOverride	Non supportato (funzionalità di richiesta)	Non supportato (funzionalità di richiesta)
	com.amazonaws.regions.RegionUtils.disableRemote	Non supportato (funzionalità di richiesta)	Non supportato (funzionalità di richiesta)

1.x Variabile di ambiente	1.x Proprietà del sistema	Variabile d'ambiente 2.x	Proprietà di sistema 2.x
	`com.amazonaws.services.s3.disableImplicitGlobalClients`	Non supportato (funzionalità di richiesta)	Non supportato (funzionalità di richiesta)
	`com.amazonaws.sdk.enableInRegionOptimizedMode`	Non supportato (funzionalità di richiesta)	Non supportato (funzionalità di richiesta)

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	`<dependencyManagement> <dependencies> <dependency> <groupId> com.amazonaws</groupId>`	`<dependencyManagement> <dependencies> <dependency> <groupId> software.amazon.awssdk</groupId> <artifactId>bom</artifactId>`

Modifica	v1	v2
	<artifact Id>aws-java-sdk-bom</ artifactId> <version> 1.12.680[1]</version> <type>pom</ type> <scope>im port</scope> </dependency> </dependencies> </dependencyManageme nt> <dependencies> <dependency> <groupId> com.amazonaws</gro upId> <artifact Id>dynamodb</artif actId> </dependency> </dependencies>	<version> 2.27.10[2]</version> <type>pom</ type> <scope>im port</scope> </dependency> </dependencies> </dependencyManageme nt> <dependencies> <dependency> <groupId> software.amazon.aw ssdk</groupId> <artifact Id>dynamodb</artif actId> </dependency> </dependencies>
Nome pacchetto	com.amazonaws.serv ices.dynamodbv2.wa iters	software.amazon.aw ssdk.services.dyna modb.waiters
Nomi delle classi	AmazonDynamoDBWait ers	• Sincrono: DynamoDbW aiter • Asincrono: DynamoDbA syncWaiter

1 Ultima versione . 2 Ultima versione.

Modifiche alle API

Modifica	v1	v2
Crea un cameriere	`AmazonDynamoDB client = AmazonDynamoDBClientBuilder .standard().build(); AmazonDynamoDBWaiters waiter = client.waiters();`	Sincrono: `DynamoDbClient client = DynamoDbClient.create(); DynamoDbWaiter waiter = client.waiter();` Asincrono: `DynamoDbAsyncClient asyncClient = DynamoDbAsyncClient.create(); DynamoDbAsyncWaiter waiter = asyncClient.waiter();`
Attendi che esista una tabella	Sincrono: `waiter.tableExists() .run(new WaiterParameters<>( new DescribeTableRequest(tableName)));` Asincrono: `waiter.tableExists() .runAsync(new WaiterParameters() .withRequest(new DescribeTableRequest(tableName)),`	Sincrono: `WaiterResponse<DescribeTableResponse> waiterResponse = waiter.waitUntilTableExists( r -> r.tableName("myTable")); waiterResponse.matched().response() .ifPresent(System.out::println);` Asincrono:

Modifica	v1	v2
	`new WaiterHandler() { @Override public void onWaitSuccess( AmazonWebServiceRequest amazonWebServiceRequest) { System.out.println("Table creation succeeded"); } @Override public void onWaitFailure(Exception e) { e.printStackTrace(); } }).get();`	`waiter.waitUntilTableExists(r -> r.tableName(tableName)) .whenComplete((r, t) -> { if (t != null) { t.printStackTrace(); } else { System.out.println( "Table creation succeeded"); } }).join();`

Modifiche di configurazione

Modifica	v1	v2

Strategia di sondaggio (numero massimo di tentativi e ritardo fisso)	`MaxAttemptsRetryStrategy maxAttemptsRetryStrategy = new MaxAttemptsRetryStrategy(10);`
	
	`FixedDelayStrategy fixedDelayStrategy = new FixedDelayStrategy(3);`
		`FixedDelayBackoffStrategy fixedDelayBackoffStrategy = FixedDelayBackoffStrategy .create(Duration.ofSeconds(3));`

Modifica	v1	v2
	PollingStrategy pollingStrategy = new PollingStrategy(maxAttempts RetryStrategy, fixedDelayStrategy); waiter.tableExists().run(new WaiterParameters<>(new DescribeTableRequest(tableName)), pollingStrategy);	waiter.waitUntilTableExists(r -> r.tableName(tableName), c -> c.maxAttempts(10) .backoffStrategy(fixedDelayBackoffStrategy));

Modifiche nell'utilità EC2 dei metadati 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	<dependencyManagement> <dependencies> <dependency> <groupId>com.amazonaws</groupId> <artifactId>aws-java-sdk-bom</artifactId> <version>*1.12.587*[1]</version>	<dependencyManagement> <dependencies> <dependency> <groupId>software.amazon.awssdk</groupId> <artifactId>bom</artifactId> <version>*2.27.21*[2]</version> <type>pom</type>

Modifica	v1	v2
	<type>pom</type> <scope>import</scope> </dependency> </dependencies> </dependencyManagement> <dependencies> <dependency> <groupId>com.amazonaws</groupId> <artifactId>aws-java-sdk-core</artifactId> </dependency> </dependencies>	<scope>import</scope> </dependency> </dependencies> </dependencyManagement> <dependencies> <dependency> <groupId>software.amazon.awssdk</groupId> <artifactId>imds</artifactId> </dependency> <dependency> <groupId>software.amazon.awssdk</groupId> <artifactId>apache-client [3]</artifactId> </dependency> </dependencies>
Nome pacchetto	`com.amazonaws.util`	`software.amazon.awssdk.imds`

Modifica	v1	v2

Approccio all'istanziazione	Utilizza metodi di utilità statici; nessuna istanziazione:
	`String localHostName = EC2MetadataUtils.getLocalHostName();`
		Usa un metodo di fabbrica statico:
		`Ec2MetadataClient client = Ec2MetadataClient.create();`
		Oppure usa un approccio da costruttore:
		`Ec2MetadataClient client = Ec2MetadataClient.builder() .endpointMode(EndpointMode.IPV6) .build();`

Tipi di clienti	Metodi di utilità solo sincroni: `EC2MetadataUtils`	Sincrono: `Ec2MetadataClient` Asincrono: `Ec2MetadataAsyncClient`

1 Ultima versione . 2 Ultima versione.

3 Notate la dichiarazione del `apache-client` modulo per la v2. La versione 2 dell'utilità per i EC2 metadati richiede un'implementazione dell'`SdkHttpClient`interfaccia per il client di metadati sincrono o dell'`SdkAsyncHttpClient`interfaccia per il client di metadati asincrono. 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 EC2 risorsa come parametro nella v2. La tabella seguente mostra i diversi approcci.

v1	v2
`String userMetaData = EC2Metadata Utils.getUserData();`	`Ec2MetadataClient client = Ec2Metada taClient.create(); Ec2MetadataResponse response = client.get("/latest/ user-data"); String userMetaData = response.asString();`

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

Attivo 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	`InstanceInfo instanceI nfo =`	`Ec2MetadataResponse response =`

Caso d'uso	v1	v2

	`EC2MetadataUtils.getInstanceInfo();`
	`String region = instanceInfo.getRegion();`
		`client.get("/latest/dynamic/instance-identity/document");`
		`Document instanceInfo = response.asDocument();`
		`String region = instanceInfo.asMap().get("region").asString();`

Recupera l'id dell'istanza	`InstanceInfo instanceInfo = EC2MetadataUtils.getInstanceInfo();`
	`String instanceId = instanceInfo.instanceId;`
		`Ec2MetadataResponse response = client.get("/latest/dynamic/instance-identity/document");`
		`Document instanceInfo = response.asDocument();`
		`String instanceId = instanceInfo.asMap().get("instanceId").asString();`

Recupera il tipo di istanza	`InstanceInfo instanceInfo = EC2MetadataUtils.getInstanceInfo();`
	`String instanceType = instanceInfo.instanceType();`
		`Ec2MetadataResponse response = client.get("/latest/dynamic/instance-identity/document");`
		`Document instanceInfo = response.asDocument();`
		`String instanceType = instanceInfo.asMap().get("instanceType").asString();`

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: `com.amazonaws.sdk.ec2MetadataServiceEndpointOverride`	Metodo di configurazione di Client Builder: `endpoint(...)`
Variabile d'ambiente: `AWS_EC2_METADATA_SERVICE_ENDPOINT`	Proprietà del sistema: `aws.ec2MetadataServiceEndpoint`
Valore predefinito: `http://169.254.169.254`	File di configurazione: `~.aws/config` con l'impostazione `ec2_metadata_service_endpoint`
	Valore associato a risolto `endpoint-mode`
	Valore predefinito: `http://169.254.169.254`

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, puoi specificare una modalità endpoint per configurare il client di metadati in modo che utilizzi i valori di endpoint predefiniti per o. IPv4 IPv6 La modalità endpoint non è disponibile per la versione 1. Il valore predefinito utilizzato per è e for IPv4 . `http://169.254.169.254` `http://[fd00:ec2::254]` IPv6

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: `endpointMode(...)`	`Ec2MetadataClient client = Ec2MetadataClient.builder().endpointMode(EndpointMode.IPV4).build();`	`EndpointMode.IPV4` , `EndpointMode.IPV6`
Proprietà del sistema	`aws.ec2MetadataServiceEndpointMode`	`IPv4`, `IPv6` (il caso non ha importanza)
File di configurazione: `~.aws/config`	`ec2_metadata_service_endpoint` Impostazione	`IPv4`, `IPv6` (il caso non ha importanza)
Non specificato nei modi precedenti	IPv4 è usato	

Come si risolve **`endpoint`** l'SDK o nella v2 **`endpoint-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` `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 la presenza di un valore dell'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 di. IPv4 `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 Version 2 ()IMDSv2: approccio orientato alla sessione

La tabella seguente confronta il modo in cui Java SDKs funziona 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 indietro 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 IMDSv2 solo 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 `retryPolicy(...)`
HTTP	Timeout di connessio ne configurabile tramite	Configurazione disponibile passando un client HTTP al

Configurazione	v1	v2
	la variabile di ambiente. `AWS_METADATA_SERVICE_TIMEOUT` L'impostazione predefinita è 1 secondo.	metodo `httpClient(...)` 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	`<dependencyManagement> <dependencies> <dependency> <groupId>com.amazonaws</groupId> <artifactId>aws-java-sdk-bom</artifactId>`	`<dependencyManagement> <dependencies> <dependency> <groupId>software.amazon.awssdk</groupId> <artifactId>bom</artifactId>`

Modifica	v1	v2
	<version> 1.12.587[1]</version> <type>pom</type> <scope>import</scope> </dependency> </dependencies> </dependencyManagement> <dependencies> <dependency> <groupId>com.amazonaws</groupId> <artifactId>cloudfront</artifactId> </dependency> </dependencies>	<version> 2.27.21[2]</version> <type>pom</type> <scope>import</scope> </dependency> </dependencies> </dependencyManagement> <dependencies> <dependency> <groupId>software.amazon.awssdk</groupId> <artifactId>cloudfront</artifactId> </dependency> </dependencies>
Nome pacchetto	`com.amazonaws.services.cloudfront`	`software.amazon.awssdk.services.cloudfront`
Nomi delle classi	CloudFrontUrlSigner CloudFrontCookieSigner	CloudFrontUtilities SignedUrl CannedSignerRequest CustomSignerRequest

[1] Ultima versione. [2] Ultima versione.

Modifiche alle API

Comportamento	v1	v2
Crea una richiesta predefinita	Gli argomenti vengono passati direttamente all'API.	`CannedSignerRequest cannedRequest = CannedSignerRequest.builder() .resourceUrl(resourceUrl) .privateKey(privateKey) .keyPairId(keyPairId) .expirationDate(expirationDate) .build();`
Crea una richiesta personalizzata	Gli argomenti vengono passati direttamente all'API.	`CustomSignerRequest customRequest = CustomSignerRequest.builder() .resourceUrl(resourceUrl) .resourceUrlPattern(resourceUrlPattern) .privateKey(keyFile) .keyPairId(keyPairId) .expirationDate(expirationDate) .activeDate(activeDate)`

Comportamento	v1	v2

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

Genera un URL firmato (predefinito)	`String signedUrl = CloudFrontUrlSigner.getSignedURLWithCannedPolicy( resourceUrl, keyPairId, privateKey, expirationDate);`	`CloudFrontUtilities cloudFrontUtilities = CloudFrontUtilities.create();`
		
		`SignedUrl signedUrl =`
		
		`cloudFrontUtilities.getSignedUrlWithCannedPolicy(cannedRequest);`
		
		`String url = signedUrl.url();`

Genera un cookie firmato (personalizzato)	`CookiesForCustomPolicy cookies = CloudFrontCookieSigner.getCookiesForCustomPolicy( resourceUrl, privateKey, keyPairId, expirationDate, activeDate, ipRange);`	`CloudFrontUtilities cloudFrontUtilities = CloudFrontUtilities.create();`
		
		`CookiesForCustomPolicy cookies = cloudFrontUtilities.getCookiesForCustomPolicy(customRequest);`

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'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	<dependencyManagement> <dependencies> <dependency> <groupId> com.amazonaws</gro upId> <artifact Id>aws-java-sdk-bom</ artifactId> <version> *1.12.587*[1]</version> <type>pom</ type> <scope>im port</scope> </dependency> </dependencies> </dependencyManageme nt> <dependencies> <dependency> <groupId> com.amazonaws</gro upId>	<dependencyManagement> <dependencies> <dependency> <groupId> software.amazon.aw ssdk</groupId> <artifact Id>bom</artifactId> <version> *2.27.21*[2]</version> <type>pom</ type> <scope>im port</scope> </dependency> </dependencies> </dependencyManageme nt> <dependencies> <dependency> <groupId> software.amazon.aw ssdk</groupId> <artifact Id>iam-policy-buil der</artifactId>

Modifica	v1	v2
	<artifact Id>aws-java-sdk-core</artifactId> </dependency> </dependencies>	</dependency> </dependencies>
Nome pacchetto	com.amazonaws.auth.policy	software.amazon.awssdk.policybuilder.iam
Nomi delle classi	Policy Statement • Dichiarazione. Effetto • IdentiyManagementActions • Risorsa • Principale • Condition	IamPolicy IamStatement • IamEffect • IamAction • IamResource • IamPrincipal • IamCondition • IamConditionOperator • IamConditionKey

1 Ultima versione . 2 Ultima versione.

Modifiche alle API

Impostazione	v1	v2
Crea un'istanza di una politica	`Policy policy = new Policy();`	`IamPolicy.Builder policyBuilder = IamPolicy.builder(); ... IamPolicy policy = policyBuilder.build();`

Impostazione	v1	v2
Imposta id	`policy.withtId(...);` `policy.setId(...);`	`policyBuilder.id(...);`
Impostare la versione	N/A: utilizza la versione predefinita di `2012-10-17`	`policyBuilder.version(...);`
Crea una dichiarazione	`Statement statement = new Statement (Effect.Allow) .withActions(...) .withConditions(...) .withId(...) .withPrincipals(...) .withResources(...);`	`IamStatement statement = IamStatement.builder() .effect(IamEffect.ALLOW) .actions(...) .notActions(...) .conditions(...) .sid(...) .principals(...) .notPrincipals(...) .resources(...) .notResources(...) .build()`
Imposta dichiarazione	`policy.withStatements(statement);` `policy.setStatements(statement);`	`policyBuilder.addStatement(statement);`

Differenze nella creazione di una dichiarazione

Operazioni

v1

L'SDK v1 dispone enumdi 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 for`SQSActions`.

```
Action action = SQSActions.SendMessage;
```

Non è possibile specificare un *NotAction* elemento in un'istruzione in v1.

v2

Nella v2, l'IamActioninterfaccia 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.

- S3BucketResource
- S3ObjectResource
- SQSQueueRisorsa

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);
```

Modifiche nella APIs mappatura/documento di DynamoDB dalla versione 1 alla versione 2

Questo argomento descrive in dettaglio le modifiche di alto livello dell'SDK Java APIs per Amazon DynamoDB dalla versione 1.x (v1) alla (v2). AWS SDK for Java 2.x Per prima cosa trattiamo l'API di object-to-table mappatura e poi discutiamo dell'API dei documenti per lavorare con documenti in stile JSON.

Modifiche di alto livello

I nomi del client di mappatura in ogni libreria differiscono in v1 e v2:

- v1 - Dinamo DBMapper
- v2 - Client avanzato per DynamoDB

Interagisci con le due librerie più o meno allo stesso modo: crei un'istanza di un mapper/client e poi fornisci un POJO Java a chi legge e scrive questi elementi nelle APIs tabelle DynamoDB. Entrambe le librerie offrono anche annotazioni per la classe del POJO per indicare il modo in cui il client gestisce il POJO.

Le differenze notevoli quando si passa alla v2 includono:

- V2 e v1 utilizzano nomi di metodi diversi per le operazioni DynamoDB di basso livello. Per esempio:

v1	v2
caricare	getItem
save	putItem

v1	v2
batchLoad	batchGetItem

- La V2 offre diversi modi per definire gli schemi delle tabelle e mapparli alle tabelle. POJOs Puoi scegliere tra l'uso di annotazioni o uno schema generato dal codice utilizzando un builder. V2 offre anche versioni mutabili e immutabili degli schemi.
- Con v2, si crea specificamente lo schema della tabella come uno dei primi passaggi, mentre nella v1, lo schema della tabella viene dedotto dalla classe annotata secondo necessità.
- La V2 include il client Document API nell'API client avanzata, mentre la v1 utilizza un'API separata.
- Tutti APIs sono disponibili nelle versioni sincrone e asincrone nella v2.

Consulta la sezione sulla mappatura di DynamoDB in questa guida per informazioni più dettagliate sul client avanzato v2.

Importa le dipendenze

v1

```
<dependencyManagement>
  <dependencies>
    <dependency>
      <groupId>com.amazonaws</gro
upId>
      <artifactId>aws-java-sdk-bom</
artifactId>
      <version> 1.X.X</version>
      <type>pom</type>
      <scope>import</scope>
    </dependency>
  </dependencies>
</dependencyManagement>

<dependencies>
  <dependency>
    <groupId>com.amazonaws</groupId>
    <artifactId>aws-java-sdk-dy
namodb</artifactId>
  </dependency>
```

v2

```
<dependencyManagement>
  <dependencies>
    <dependency>
      <groupId>software.amazon.aw
ssdk</groupId>
      <artifactId>bom</artifactId>
      <version> 2.X.X*</version>
      <type>pom</type>
      <scope>import</scope>
    </dependency>
  </dependencies>
</dependencyManagement>

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

v1	v2
`</dependencies>`	`</dependencies>`

* Ultima versione.

Nella v1, una singola dipendenza include sia l'API DynamoDB di basso livello che l'API mapping/document, mentre nella v2, si utilizza la dipendenza artifact per accedere all'API di mappatura/documento. `dynamodb-enhanced` Il modulo contiene una dipendenza transitiva dal modulo di `dynamodb` basso livello. `dynamodb-enhanced`

Modifiche all'API

Crea un cliente

Caso d'uso	v1	v2
Istanziazione normale	`AmazonDynamoDB standardClient = AmazonDynamoDBClientBuilder.standard() .withCredentials(credentialsProvider) .withRegion(Regions.US_EAST_1) .build(); DynamoDBMapper mapper = new DynamoDBMapper(standardClient);`	`DynamoDbClient standardClient = DynamoDbClient.builder() .credentialsProvider(ProfileCredentialsProvider.create()) .region(Region.US_EAST_1) .build(); DynamoDbEnhancedClient enhancedClient = DynamoDbEnhancedClient.builder() .dynamoDbClient(standardClient) .build();`
Istanziazione minima	`AmazonDynamoDB standardClient = AmazonDynamoDBClientBuilder.standard();`	`DynamoDbEnhancedClient enhancedClient = DynamoDbEnhancedClient.create();`

Caso d'uso	v1	v2
	`DynamoDBMapper mapper = new DynamoDBMapper(standardClient);`	
Con trasformatore di attributi *	`DynamoDBMapper mapper = new DynamoDBMapper(standardClient, attributeTransformerInstance);`	`DynamoDbEnhancedClient enhancedClient = DynamoDbEnhancedClient.builder() .dynamoDbClient(standardClient) .extensions(extensionAInstance, extensionBInstance) .build();`

* Le estensioni in v2 corrispondono all'incirca ai trasformatori di attributi in v1. La the section called "Usa le estensioni" sezione contiene ulteriori informazioni sulle estensioni in v2.

Stabilire la mappatura alla tabella/indice DynamoDB

Nella v1, si specifica il nome di una tabella DynamoDB tramite un'annotazione bean. Nella v2, un metodo factory`table()`, produce un'istanza `DynamoDbTable` che rappresenta la tabella DynamoDB remota. Il primo parametro del `table()` metodo è il nome della tabella DynamoDB.

Caso d'uso	v1	v2
Mappa la classe Java POJO sulla tabella DynamoDB	`@DynamoDBTable(tableName ="Customer") public class Customer { ... }`	`DynamoDbTable<Customer> customerTable = enhancedClient.table("Customer", TableSchema.fromBean(Customer.class));`
Mappare su un indice secondario DynamoDB	1. Definire una classe POJO che rappresenti l'indice.	1. Annota gli attributi di una classe POJO con `@DynamoDbSecondaryPartitionKey` (per

Caso d'uso	v1	v2

	• Annota la classe `@DynamoDBTable` fornendo il nome della tabella che contiene l'indice.
	• Annota le proprietà con `@DynamoDBIndexHashKey` e facoltativamente. `@DynamoDBIndexRangeKey`
	2. Crea un'espressione di interrogazione.
	3. Interrogazione utilizzando il riferimento alla classe POJO che rappresenta l'indice. Ad esempio
	`mapper.query(IdEmailIndex.class, queryExpression)`
	dove `IdEmailIndex` è la classe di mappatura per l'indice.
	
	La sezione della DynamoDB Developer Guide che illustra il metodo `query` v1 mostra un esempio completo.
		un GSI) e `@DynamoDbSecondarySortKey` (per un GSI o LSI). Ad esempio,
		`@DynamoDbSecondarySortKey(indexNames = "IdEmailIndex") public String getEmail() { return this.email; }`
		2. Recuperate un riferimento all'indice. Ad esempio,
		`DynamoDbIndex<Customer> customerIndex = customerTable.index("IdEmailIndex");`
		3. Interroga l'indice.
		
		La ??? sezione di questa guida fornisce ulteriori informazioni.

Operazioni sulle tabelle

Questa sezione descrive le operazioni APIs che differiscono tra v1 e v2 per la maggior parte dei casi d'uso standard.

Nella v2, tutte le operazioni che coinvolgono una singola tabella vengono chiamate sull'`DynamoDbTable`istanza, non sul client avanzato. Il client avanzato contiene metodi che possono indirizzare più tabelle.

Nella tabella denominata Table operations riportata di seguito, un'istanza POJO viene definita come `item` o come un tipo specifico come`customer1`. Per gli esempi della versione 2, le istanze denominate `table` sono il risultato di una chiamata precedente `enhancedClient.table()` che restituisce un riferimento all'istanza. `DynamoDbTable`

Tieni presente che la maggior parte delle operazioni v2 può essere richiamata con uno schema di consumo fluido anche se non vengono visualizzate. Ad esempio,

```
Customer customer = table.getItem(r # r.key(key));
  or
Customer customer = table.getItem(r # r.key(k ->
 k.partitionValue("id").sortValue("email")))
```

Per le operazioni v1, la tabella contiene alcuni dei moduli più utilizzati e non tutti i moduli sovraccaricati. Ad esempio, il `load()` metodo presenta i seguenti sovraccarichi:

```
mapper.load(Customer.class, hashKey)
mapper.load(Customer.class, hashKey, rangeKey)
mapper.load(Customer.class, hashKey, config)
mapper.load(Customer.class, hashKey, rangeKey, config)
mapper.load(item)
mapper.load(item, config)
```

La tabella mostra i moduli più comunemente usati:

```
mapper.load(item)
mapper.load(item, config)
```

Operazioni sulle tabelle

Caso d'uso	v1	Funzionamento Dynam	v2

Scriver un Java POJO su una tabella Dynam	`mapper.save(item)`
	`mapper.save(item, config)`
	`mapper.save(item, saveExpression, config)`
	
	Nella v1, `DynamoDBMapperConfig.SaveBehavior` and annotations determina quale metodo DynamoDB di basso livello verrà chiamato. In generale, `UpdateItem` viene chiamato tranne quando si usa and. `SaveBehavior.CLOBBER` `SaveBehavior.PUT` Le chiavi generate automaticamente sono un caso d'uso speciale e occasionalmente `UpdateItem` vengono utilizzate entrambe`PutItem`.
		`PutItem` Update m	`table.putItem(putItemRequest)`
			`table.putItem(item)`
			`table.putItemWithResponse(item) // Returns metadata.`
			
			`updateItem(updateItemRequest)`
			`table.updateItem(item)`
			`table.updateItemWithResponse(item) //Returns metadata.`

Legger un elemen da una tabella Dynam a un	`mapper.load(item)`
	`mapper.load(item, config)`
		`GetIt`	`table.getItem(getItemRequest)`
			`table.getItem(item)`
			`table.getItem(key)`
			`table.getItemWithResponse(key) // Returns POJO with metadata.`

Caso d'uso	v1	Funzionamento Dynam	v2
Java POJO			

Elimina un elemer da una tabella Dynam	`mapper.delete(item , deleteExpression, config)`	`Delet m`	`table.deleteItem(deleteItemRequest)`
			`table.deleteItem(item)`
			`table.deleteItem(key)`

Interro una tabella o un indice secondo Dynam e restitui ci un elenco impagi o	`mapper.query(Custo mer.class, queryExpr ession)`
	`mapper.query(Custom er.class, queryExpr ession, mapperConfig)`
		`Query`	`table.query(queryRequest)`
			`table.query(queryConditional)`
			
			Utilizza il valore restituito `PageItera ble.stream()` (lazy loading) per le risposte di sincronizzazione e per le risposte asincrone `PagePublisher.subs cribe()`

Caso d'uso	v1	Funzionamento Dynam	v2

Interrogare una tabella o un indice secondario Dynamo e restituisci un elenco	`mapper.queryPage(Customer.class, queryExpression)`
	`mapper.queryPage(Customer.class, queryExpression, mapperConfig)`
		`Query`	`table.query(queryRequest)`
			`table.query(queryConditional)`
			
			Utilizza il valore restituito `PageIterable.items()` (lazy loading) per le risposte di sincronizzazione e `PagePublisher.items.subscribe()` per le risposte asincrone

Scansiona una tabella o un indice secondario Dynamo e restituisci un elenco impagino	`mapper.scan(Customer.class, scanExpression)`
	`mapper.scan(Customer.class, scanExpression, mapperConfig)`
		`Scan`	`table.scan()`
			`table.scan(scanRequest)`
			
			Utilizza il valore restituito `PageIterable.stream()` (lazy loading) per le risposte di sincronizzazione e per le risposte asincrone `PagePublisher.subscribe()`

Caso d'uso	v1	Funzionamento Dynam	v2

Scansiona una tabella o un indice secondario Dynamo e restituisci un elenco	`mapper.scanPage(Customer.class, scanExpression)`
	`mapper.scanPage(Customer.class, scanExpression, mapperConfig)`
		`Scan`	`table.scan()`
			`table.scan(scanRequest)`
			
			Utilizza il valore restituito `PageIterable.items()` (lazy loading) per le risposte di sincronizzazione e `PagePublisher.items.subscribe()` per le risposte asincrone

Caso d'uso	v1	Funzionamento DynamoDB	v2

Leggi più elementi da più tabelle in un batch	`mapper.batchLoad(Arrays.asList(customer1, customer2, book1))`
	`mapper.batchLoad(itemsToGet) // itemsToGet: Map<Class<?>, List<KeyPair>>`
		BatchGetItem	`enhancedClient.batchGetItem(batchGetItemRequest)`
			
			`enhancedClient.batchGetItem(r -> r.readBatches(`
			`ReadBatch.builder(Record1.class)`
			`.mappedTableResource(mappedTable1)`
			`.addGetItem(i -> i.key(k -> k.partitionValue(0)))`
			`.build(),`
			`ReadBatch.builder(Record2.class)`
			`.mappedTableResource(mappedTable2)`
			`.addGetItem(i -> i.key(k -> k.partitionValue(0)))`
			`.build()))`
			
			`// Iterate over pages with lazy loading or over all items from the same table.`

Caso d'uso	v1	Funzionamento DynamoDB	v2

Scrivi più elementi su più tabelle in un batch	`mapper.batchSave(Arrays.asList(customer1, customer2, book1))`	BatchWriteItem	`enhancedClient.batchWriteItem(batchWriteItemRequest)`
			
			`enhancedClient.batchWriteItem(r -> r.writeBatches(`
			`WriteBatch.builder(Record1.class)`
			`.mappedTableResource(mappedTable1)`
			`.addPutItem(item1)`
			`.build(),`
			`WriteBatch.builder(Record2.class)`
			`.mappedTableResource(mappedTable2)`
			`.addPutItem(item2)`
			`.build()))`

Elimina più elementi da più tabelle in un batch	`mapper.batchDelete(Arrays.asList(customer1, customer2, book1))`	BatchWriteItem	`enhancedClient.batchWriteItem(r -> r.writeBatches(`
			`WriteBatch.builder(Record1.class)`
			`.mappedTableResource(mappedTable1)`
			`.addDeleteItem(item1key)`
			`.build(),`
			`WriteBatch.builder(Record2.class)`
			`.mappedTableResource(mappedTable2)`
			`.addDeleteItem(item2key)`
			`.build()))`

Caso d'uso	v1	Funzionamento Dynam	v2
Scrittura / elimina più elemer in un batch	`mapper.batchWrite( Arrays.asList(cust omer1, book1), Arrays.as List(customer2))`	BatchW eItem	`enhancedClient.batchWriteItem(r -> r.writeBatches( WriteBatch.builder(Record1. class) .mappedTableResour ce(mappedTable1) .addPutItem(item1) .build(), WriteBatch.builder(Record2. class) .mappedTableResour ce(mappedTable2) .addDeleteItem(ite m2key) .build()))`
Esegui una scrittur transaz nale	`mapper.transaction Write(transactionW riteRequest)`	TransW riteI	`enhancedClient.transactWrit eItems(transasctWriteItemsRequest)`
Effettu e una lettura transaz nale	`mapper.transaction Load(transactionLo adRequest)`	TransG etIte	`enhancedClient.transactGetI tems(transactGetItemsRequest)`

Caso d'uso	v1	Funzionamento Dynam	v2
Ottieni il conteg degli elemer corrisp denti di una scansi o di una query	`mapper.count(Customer.class, queryExpression)` `mapper.count(Customer.class, scanExpression)`	`Query` `Scan` con `Selec` `UNT`	Non supportato

Crea una tabella in Dynam corrisp dente alla classe POJO	`mapper.generateCreateTableRequest(Customer.class)`
	L'istruzione precedente genera una richiesta di creazione della tabella di basso livello; gli utenti devono chiamare `createTable` il client DynamoDB.
		Create le	`table.createTable(createTableRequest)`
			
			`table.createTable(r -> r.provisionedThroughput(getDefaultProvisionedThroughput())`
			`.globalSecondaryIndices(`
			`EnhancedGlobalSecondaryIndex.builder()`
			`.indexName("gsi_1")`
			`.projection(p -> p.projectionType(ProjectionType.ALL))`
			`.provisionedThroughput(getDefaultProvisionedThroughput())`
			`.build()));`

Caso d'uso	v1	Funzionamento Dynam	v2
Esegui una scansi parall in Dynam	`mapper.parallelScan(Customer.class, scanExpression, numTotalSegments)`	`Scanc` e parame `Segme` `Total` ents	Gli utenti sono tenuti a gestire i thread di lavoro e a chiamare `scan` per ogni segmento: `table.scan(r -> r.segment(0).totalSegments(5))`
Integra Amazc S3 con Dynam per archivi e collega nti S3 intellig nti	`mapper.createS3Link(bucket, key)` `mapper.getS3ClientCache()`	-	Non è supportato perché abbina Amazon S3 e DynamoDB.

Classi e proprietà della mappa

Sia in v1 che in v2, si mappano le classi alle tabelle utilizzando annotazioni in stile bean. V2 offre anche altri modi per definire schemi per casi d'uso specifici, come lavorare con classi immutabili.

Annotazioni Bean

La tabella seguente mostra le annotazioni bean equivalenti per un caso d'uso specifico utilizzate in v1 e v2. Uno scenario `Customer` di classe viene utilizzato per illustrare i parametri.

Le annotazioni, così come le classi e le enumerazioni, nella v2 seguono la convenzione camel case e utilizzano '', non 'DynamoDB'. DynamoDb

Caso d'uso	v1	v2
Mappa la classe sulla tabella	`@DynamoDBTable (tableName ="CustomerTable")`	`@DynamoDbBean` `@DynamoDbBean(converterProviders = {...})` Il nome della tabella viene definito quando si chiama il `DynamoDbEDnhancedClient#table()` metodo.
Designate un membro della classe come attributo della tabella	`@DynamoDBAttribute (attributeName = "customerName")`	`@DynamoDbAttribute("customerName")`
Designare un membro della classe come chiave hash/di partizione	`@DynamoDBHashKey`	`@DynamoDbPartitionKey`
Designare un membro della classe è una chiave di intervallo/ ordinamento	`@DynamoDBRangeKey`	`@DynamoDbSortKey`
Designare un membro della classe come chiave di hash/ partizione	`@DynamoDBIndexHash Key`	`@DynamoDbSecondaryPartitionKey`

Caso d'uso	v1	v2
dell'indice secondaria		
Designare un membro della classe come chiave di ordinamento/intervallo di indice secondaria	`@DynamoDBIndexRangeKey`	`@DynamoDbSecondarySortKey`
Ignora questo membro della classe durante la mappatura su una tabella	`@DynamoDBIgnore`	`@DynamoDbIgnore`
Designare un membro della classe come attributo chiave UUID generato automaticamente	`@DynamoDBAutoGeneratedKey`	`@DynamoDbAutoGeneratedUuid` L'estensione che lo fornisce non viene caricata per impostazione predefinita; è necessario aggiungere l'estensione a client builder.

Caso d'uso	v1	v2
Designare un membro della classe come attributo timestamp generato automatic amente	`@DynamoDBAutoGeneratedTimestamp`	`@DynamoDbAutoGeneratedTimestampAttribute` L'estensione che lo fornisce non viene caricata per impostazione predefinita; è necessario aggiungere l'estensione a client builder.
Designare un membro della classe come attributo di versione con incremento automatico	`@DynamoDBVersionAttribute`	`@DynamoDbVersionAttribute` L'estensione che lo fornisce viene caricata automatic amente.
Designate un membro della classe come richieden te una conversio ne personali zzata	`@DynamoDBTypeConverted`	`@DynamoDbConvertedBy`

Caso d'uso	v1	v2
Designate un membro della classe da memorizza re come tipo di attributo diverso	`@DynamoDBTyped(<Dy namoDBAttributeTyp e>)`	Nessun equivalente
Designare una classe che può essere serializz ata in un documento DynamoDB (document o in stile JSON) o in un documento secondario	`@DynamoDBDocument`	Nessuna annotazione equivalente diretta. Utilizza l'API Enhanced Document.

Annotazioni aggiuntive V2

Caso d'uso	v1	v2
Designate un membro della classe da non memorizza re come attributo NULL se il valore Java è nullo	N/D	`@DynamoDbIgnoreNulls`

Caso d'uso	v1	v2
Designate un membro della classe come oggetto vuoto se tutti gli attributi sono nulli	N/D	`@DynamoDbPreserveEmptyObject`
Designate un'azione speciale di aggiornamento per un membro della classe	N/D	`@DynamoDbUpdateBehavior`
Designate una classe immutabile	N/D	`@DynamoDbImmutable`
Designare un membro della classe come attributo del contatore ad incremento automatico	N/D	`@DynamoDbAtomicCounter` L'estensione che fornisce questa funzionalità viene caricata automaticamente.

Configurazione

Nella v1, in genere si controllano comportamenti specifici utilizzando un'istanza di. `DynamoDBMapperConfig` È possibile fornire l'oggetto di configurazione quando si crea il mapper o quando si effettua una richiesta. Nella v2, la configurazione è specifica dell'oggetto di richiesta per l'operazione.

Caso d'uso	v1	Impostazione predefinita in v1	v2
	`DynamoDBMapperConfig.builder()`		
Strategia di nuovi	`.withBatchLoadRetryStrategy`	riprova gli	

Caso d'uso	v1	Imposta one predefin ta in v1	v2
tentativi di caricamer to in batch	`(batchLoadRetryStr ategy)`	elementi non riusciti	
Strategia di ripetizio ne dei tentativi di scrittura in batch	`.withBatc hWriteRetryStrateg y(batchWriteRetryS trategy)`	riprova gli elementi non riusciti	
Letture coerenti	`.withCons istentReads(CONSIS TENT)`	EVENTU	Per impostazione predefinita, le letture coerenti sono false per le operazioni di lettura. Sostituisci con `.consistentRead(true)` sull'oggetto della richiesta.
Schema di conversio ne con set di marshalle rs/ unmars hallers	`.withConv ersionSchema(conve rsionSchema)` Le implementazioni statiche forniscono la retrocompatibilità con le versioni precedenti.	V2_COM IBLE	Non applicabile. Si tratta di una funzionalità legacy che si riferisce al modo in cui le versioni precedenti di DynamoDB (v1) memorizzavano i tipi di dati e questo comportamento non verrà mantenuto nel client avanzato. Un esempio di comportamento in DynamoDB v1 è l'archivi azione dei valori booleani come numero anziché come booleano.

Caso d'uso	v1	Imposta one predefin ta in v1	v2
Nomi delle tabelle	`.withObje ctTableNameResolve r()` `.withTabl eNameOverride()` `.withTabl eNameResolver()` Le implementazioni statiche forniscono la retrocompatibilità con le versioni precedenti	usa annotati n o guess from class	Il nome della tabella viene definito quando si chiama il `DynamoDbEDnhancedC lient#table()` metodo.
Strategia di caricamer to dell'impa ginazione	`.withPagi nationLoadingStrat egy(strategy)` Le opzioni sono: `LAZY_LOADING`, o `EAGER_LOADING` `ITERATION_ONLY`	`LAZY_L ING`	L'impostazione predefinita è solo l'iterazione. Le altre opzioni v1 non sono supportate.
Richiedi la raccolta delle metriche	`.withRequ estMetricCollector (collector)`	`null`	Da utilizzare `metricPublisher()` `ClientOverrideConfiguration` durante la creazione del client DynamoDB standard.

Caso d'uso	v1	Imposta one predefin ta in v1	v2
Salva il comportar ento	`.withSave Behavior(SaveBehav ior.CLOBBER)` Le opzioni sono `UPDATECLOBBER,PUT,APPI T` , o`UPDATE_SK IP_NULL_A TTRIBUTES` .	UPDATE	Nella v2, si chiama `putItem()` or in `updateItem()` modo esplicito. `CLOBBER or PUT`: L'azione corrispondente nella v 2 è la chiamata. `putItem()` Non esiste una `CLOBBER` configurazione specifica. `UPDATE`: Corrisponde a `updateItem()` `UPDATE_SKIP_NULL_ATTRIBUTES` : Corrisponde a`updateItem()` . Controlla il comportamento di aggiornamento con l'imposta zione della richiesta `ignoreNulls` e l'annotaz ione/tag`DynamoDbUpdateBehavior` . `APPEND_SET` : non supportato
Tipo: fabbrica di convertit ori	`.withType ConverterFactory(t ypeConverterFactor y)`	convertit ori di tipo standarc	Impostare sul fagiolo utilizzando `@DynamoDbBean(converterProviders = {ConverterProvider.class, DefaultAttributeConverterPr ovider.class})`

Configurazione per operazione

Nella v1, alcune operazioni, ad esempio`query()`, sono altamente configurabili tramite un oggetto «espressione» inviato all'operazione. Per esempio:

```
DynamoDBQueryExpression<Customer> emailBwQueryExpr = new
 DynamoDBQueryExpression<Customer>()
    .withRangeKeyCondition("Email",
        new Condition()
```

```
            .withComparisonOperator(ComparisonOperator.BEGINS_WITH)
            .withAttributeValueList(
                new AttributeValue().withS("my")));

mapper.query(Customer.class, emailBwQueryExpr);
```

Nella v2, invece di utilizzare un oggetto di configurazione, si impostano i parametri sull'oggetto di richiesta utilizzando un builder. Per esempio:

```
QueryEnhancedRequest emailBw = QueryEnhancedRequest.builder()
    .queryConditional(QueryConditional
        .sortBeginsWith(kb -> kb
            .sortValue("my"))).build();

customerTable.query(emailBw);
```

Condizionali

Nella v2, le espressioni condizionali e di filtraggio vengono espresse utilizzando un `Expression` oggetto, che incapsula la condizione e la mappatura di nomi e filtri.

Caso d'uso	Operazioni	v1	v2
Condizioni previste per gli attributi	save (), delete (), query (), scan ()	`new DynamoDBSaveExpression() .withExpected(Collections.singletonMap( "otherAttribute", new ExpectedAttributeValue(false))) .withConditionalOperator(ConditionalOperator.AND);`	Obsoleto; da usare al suo posto. `ConditionExpression`
Espressione della	delete ()	`deleteExpression.setConditionExpress`	`Expression conditionExpression = Expression.builder()`

Caso d'uso	Operazioni	v1	v2

condizione		`ion("zipcode = :zipcode") deleteExpression .setExpressionAttr ibuteValues(...)`	`.expression("#key = :value OR #key1 = :value1") .putExpressionName("#key", "attribute") .putExpressionName ("#key1", "attribute3") .putExpressionValu e(":value", AttributeValues.st ringValue("wrong")) .putExpressionValu e(":value1", AttributeValues.st ringValue("three")) .build();`
			
			`DeleteItemEnhancedRequest request = DeleteItemEnhancedRequest.b uilder() .conditionExpressi on(conditionExpression).build();`

Espressione filtro	interrogazione (), scansione ()	`scanExpression .withFilterExpress ion("#statename = :state") .withExpressionAtt ributeValues(attri buteValueMapBuilde r.build()) .withExpressionAtt ributeNames(attrib uteNameMapBuilder. build())`	`Map<String, AttributeValue> values = singletonMap(":key", stringVal ue("value")); Expression filterExpression = Expression.builder() .expression("name = :key") .expressionValues(values) .build(); QueryEnhancedRequest request = QueryEnhancedRequest.builder() .filterExpression(filterExp ression).build();`

Caso d'uso	Operazioni	v1	v2

Espressione di condizione per la query	interrogazione ()	`queryExpression.withKeyConditionExpression()`	`QueryConditional keyEqual = QueryConditional.keyEqualTo(b -> b .partitionValue("movie01"));`
			
			`QueryEnhancedRequest tableQuery = QueryEnhancedRequest.builder() .queryConditional(keyEqual) .build();`

Conversione del tipo

Convertitori predefiniti

Nella v2, l'SDK fornisce un set di convertitori predefiniti per tutti i tipi più comuni. È possibile modificare i convertitori di tipo sia a livello generale del provider che per un singolo attributo. Puoi trovare un elenco dei convertitori disponibili nel riferimento AttributeConverterAPI.

Imposta un convertitore personalizzato per un attributo

Nella v1, è possibile annotare un metodo getter con `@DynamoDBTypeConverted` per specificare la classe che esegue la conversione tra il tipo di attributo Java e un tipo di attributo DynamoDB. Ad esempio, è possibile applicare una stringa `CurrencyFormatConverter` che converte tra un `Currency` tipo Java e una stringa DynamoDB come mostrato nel seguente frammento.

```
@DynamoDBTypeConverted(converter = CurrencyFormatConverter.class)
public Currency getCurrency() { return currency; }
```

L'equivalente v2 dello snippet precedente è mostrato di seguito.

```
@DynamoDbConvertedBy(CurrencyFormatConverter.class)
public Currency getCurrency() { return currency; }
```

> **Note**
>
> In v1, puoi applicare l'annotazione all'attributo stesso, a un tipo o a un'annotazione definita dall'utente, v2 supporta l'applicazione dell'annotazione solo al getter.

Aggiungi una fabbrica o un fornitore di convertitori di tipi

Nella v1, puoi fornire il tuo set di convertitori di tipo o sostituire i tipi che ti interessano aggiungendo una fabbrica di convertitori di tipi alla configurazione. Il type converter factory si estende e `DynamoDBTypeConverterFactory` le sostituzioni vengono eseguite ottenendo un riferimento al set predefinito ed estendendolo. Il seguente frammento mostra come eseguire questa operazione.

```
DynamoDBTypeConverterFactory typeConverterFactory =
    DynamoDBTypeConverterFactory.standard().override()
        .with(String.class, CustomBoolean.class, new DynamoDBTypeConverter<String,
 CustomBoolean>() {
            @Override
            public String convert(CustomBoolean bool) {
                return String.valueOf(bool.getValue());
            }
            @Override
            public CustomBoolean unconvert(String string) {
                return new CustomBoolean(Boolean.valueOf(string));
            }}).build();
DynamoDBMapperConfig config =
    DynamoDBMapperConfig.builder()
        .withTypeConverterFactory(typeConverterFactory)
        .build();
DynamoDBMapper mapperWithTypeConverterFactory = new DynamoDBMapper(dynamo, config);
```

La versione 2 offre funzionalità simili tramite l'annotazione. `@DynamoDbBean` È possibile fornire una singola `AttributeConverterProvider` o una catena di messaggi ordinati`AttributeConverterProvider`. Tieni presente che se fornisci la tua catena di fornitori di convertitori di attributi, sostituirai il fornitore di convertitori predefinito e dovrai includerlo nella catena per utilizzare i suoi convertitori di attributi.

```
@DynamoDbBean(converterProviders = {
   ConverterProvider1.class,
   ConverterProvider2.class,
```

```
   DefaultAttributeConverterProvider.class})
public class Customer {
  ...
}
```

La sezione sulla conversione degli attributi di questa guida contiene un esempio completo per la versione 2.

API dei documenti

L'API Document supporta l'utilizzo di documenti in stile JSON come elementi singoli in una tabella DynamoDB. L'API Document v1 ha un'API corrispondente nella v2, ma invece di utilizzare un client separato per l'API dei documenti come nella v1, la v2 incorpora le funzionalità dell'API dei documenti nel client avanzato DynamoDB.

Nella v1, la `Item`classe rappresenta un record non strutturato da una tabella DynamoDB. Nella v2, un record non strutturato è rappresentato da un'istanza della classe. `EnhancedDocument` Nota che le chiavi primarie sono definite nello schema della tabella per v2 e sull'elemento stesso in v1.

La tabella seguente confronta le differenze tra il Documento APIs in v1 e v2.

Caso d'uso	v1	v2
Crea un client per documenti	`AmazonDynamoDB client = ... //Create a client. DynamoDB documentClient = new DynamoDB(client);`	`// The v2 Document API uses the same DynamoDbEnhancedClient // that is used for mapping POJOs. DynamoDbClient standardClient = ... //Create a standard client. DynamoDbEnhancedClient enhancedClient = ... // Create an enhanced client.`
Fai riferimento a una tabella	`Table documentTable = docClient.documentClient("Person");`	`DynamoDbTable<EnhancedDocument> documentTable =`

Caso d'uso	v1	v2
		`enhancedClient.table("Person", TableSchema.documentSchemaBuilder() .addIndexPartitionKey(TableMetadata.primaryIndexName(),"id", AttributeValueType.S) .attributeConverterProviders(AttributeConverterProvider.defaultProvider()) .build());`
Work with semi-structured data		
Put Item	`Item item = new Item() .withPrimaryKey("id", 50) .withString("firstName", "Shirley"); PutItemOutcome outcome = documentTable.putItem(item);`	`EnhancedDocument personDocument = EnhancedDocument.builder() .putNumber("id", 50) .putString("firstName", "Shirley") .build(); documentTable.putItem(personDocument);`

Caso d'uso	v1	v2

Get Item	`GetItemOutcome outcome = documentTable.getItemOutcome( "id", 50);`
	`Item personDocFromDb = outcome.getItem();`
	`String firstName = personDocFromDb.getString("firstName");`
		`EnhancedDocument personDocFromDb = documentTable .getItem( Key.builder() .partitionValue(50) .build());`
		`String firstName = personDocFromDb.getString("firstName");`

Work with JSON items		
Converti una struttura JSON per usarla con l'API Document	`// The 'jsonPerson' identifier is a JSON string.` `Item item = new Item().fromJSON(jsonPerson);`	`// The 'jsonPerson' identifier is a JSON string.` `EnhancedDocument document = EnhancedDocument.builder() .json(jsonPerson).build());`
Inserisci JSON	`documentTable.putItem(item)`	`documentTable.putItem(document);`
Leggi JSON	`GetItemOutcome outcome = //Get item.` `String jsonPerson = outcome.getItem().toJSON();`	`String jsonPerson = documentTable.getItem(Key.builder() .partitionValue(50).build()) .fromJson();`

Riferimento all'API e guide per il documento APIs

	v1	v2
Riferimento API	Javadoc	Javadoc
Guida alla documentazione	Guida per gli sviluppatori di Amazon DynamoDB	Enhanced Document API (questa guida)

Domande frequenti

D: Il blocco ottimistico con un numero di versione funziona allo stesso modo nella v2 e nella v1?

R. Il comportamento è simile, ma la v2 no, non aggiunge automaticamente condizioni per le operazioni di eliminazione. È necessario aggiungere manualmente le espressioni di condizione se si desidera controllare il comportamento di eliminazione.

Modifiche al batching automatico delle richieste Amazon SQS dalla versione 1 alla versione 2

Questo argomento descrive in dettaglio le modifiche al batching automatico delle richieste per Amazon SQS tra la versione 1 e la versione 2 di. AWS SDK per Java

Modifiche di alto livello

La versione AWS SDK per Java 1.x esegue il buffering lato client utilizzando una `AmazonSQSBufferedAsyncClient` classe separata che richiede l'inizializzazione esplicita per il batching delle richieste.

AWS SDK for Java 2.x Semplifica e migliora la funzionalità di buffering con. `SqsAsyncBatchManager` L'implementazione di questa interfaccia fornisce funzionalità di raggruppamento automatico delle richieste direttamente integrate con lo standard. `SqsAsyncClient` Per ulteriori informazioni sulle versioni 2`SqsAsyncBatchManager`, consulta l'the section called “Utilizza il raggruppamento automatico delle richieste”argomento di questa guida.

Modifica	v1	v2
Dipendenze da Maven	<dependencyManagement> <dependencies> <dependency> <groupId> com.amazonaws</gro upId> <artifact Id>aws-java-sdk-bom</ artifactId> <version> *1.12.782*[1]</version> <type>pom</ type> <scope>im port</scope> </dependency> </dependencies> </dependencyManageme nt> <dependencies> <dependency> <groupId> com.amazonaws</gro upId> <artifact Id>aws-java-sdk-sqs</ artifactId> </dependency> </dependencies>	<dependencyManagement> <dependencies> <dependency> <groupId> software.amazon.aw ssdk</groupId> <artifact Id>bom</artifactId> <version> *2.31.15*[2]</version> <type>pom</ type> <scope>im port</scope> </dependency> </dependencies> </dependencyManageme nt> <dependencies> <dependency> <groupId> software.amazon.aw ssdk</groupId> <artifact Id>sqs</artifactId> </dependency> </dependencies>
Nomi dei pacchetti	`com.amazonaws.serv ices.sqs.buffered`	`software.amazon.aw ssdk.services.sqs. batchmanager`
Nomi delle classi	`AmazonSQSBufferedA syncClient`	`SqsAsyncBatchManager`

[1] Ultima versione. [2] Ultima versione.

Utilizzo del batching automatico delle richieste SQS

Modifica	v1	v2
Crea un gestore di batch	`AmazonSQSAsync sqsAsync = new AmazonSQSAsyncClient(); AmazonSQSAsync bufferedSqs = new AmazonSQSBufferedAsyncClient(sqsAsync);`	`SqsAsyncClient asyncClient = SqsAsyncClient.create(); SqsAsyncBatchManager sqsAsyncBatchManager = asyncClient.batchManager();`
Crea un gestore di batch con configurazione personalizzata	`AmazonSQSAsync sqsAsync = new AmazonSQSAsyncClient(); QueueBufferConfig queueBufferConfig = new QueueBufferConfig() .withMaxBatchOpenMs(200) .withMaxBatchSize(10) .withMinReceiveWaitTimeMs(1000) .withVisibilityTimeoutSeconds(20) .withReceiveMessageAttributeNames(messageAttributeValues); AmazonSQSAsync bufferedSqs = new AmazonSQSBufferedAsyncClien`	`BatchOverrideConfiguration batchOverrideConfiguration = BatchOverrideConfiguration.builder() .sendRequestFrequency(Duration.ofMillis(200)) .maxBatchSize(10) .receiveMessageMinWaitDuration(Duration.ofMillis(1000)) .receiveMessageVisibilityTimeout(Duration.ofSeconds(20)) .receiveMessageSystemAttributeNames(messageSystemAttributeNames) .receiveMessageAttributeNames(messageAttributeValues) .build();`

Modifica	v1	v2
	`t(sqsAsync, queueBuff erConfig);`	`SqsAsyncBatchManager sqsAsyncBatchManager = SqsAsyncBatchManag er.builder() .override Configuration(batc hOverrideConfigura tion) .client(S qsAsyncClient.crea te()) .schedule dExecutor(Executor s.newScheduledThre adPool(8)) .build();`
Inviare messaggi	`Future<SendMessage Result> sendResul tFuture = bufferedS qs.sendMessageAsyn c(new SendMessa geRequest() .withQueueUrl(queu eUrl) .withMessageBody(b ody));`	`CompletableFuture< SendMessageResponse> sendCompletableFuture = sqsAsyncB atchManager.sendMe ssage( SendMessageRequest .builder() .queueUrl(queueUrl) .messageBody(body) .build());`

Modifica	v1	v2
Eliminare messaggi	`Future<DeleteMessageResult> deletResultFuture = bufferedSqs.deleteMessageAsync(new DeleteMessageRequest() .withQueueUrl(queueUrl));`	`CompletableFuture<DeleteMessageResponse> deleteResultCompletableFuture = sqsAsyncBatchManager.deleteMessage( DeleteMessageRequest.builder() .queueUrl(queueUrl) .build());`
Modificare la visibilità dei messaggi	`Future<ChangeMessageVisibilityResult> changeVisibilityResultFuture = bufferedSqs.changeMessageVisibilityAsync (new ChangeMessageVisibilityRequest() .withQueueUrl(queueUrl) .withVisibilityTimeout(20));`	`CompletableFuture<ChangeMessageVisibilityResponse> changeResponseCompletableFuture = sqsAsyncBatchManager.changeMessageVisibility( ChangeMessageVisibilityRequest.builder() .queueUrl(queueUrl) .visibilityTimeout(20) .build());`

Modifica	v1	v2
Ricevere messaggi	`ReceiveMessageResult receiveResult = bufferedSqs.receiveMessage( new ReceiveMessageRequest() .withQueueUrl(queueUrl));`	`CompletableFuture<ReceiveMessageResponse> responseCompletableFuture = sqsAsyncBatchManager.receiveMessage( ReceiveMessageRequest.builder() .queueUrl(queueUrl) .build());`

Differenze tra i tipi di restituzione asincrona

Modifica	v1	v2
Tipo restituito	`Future<ResultType>`	`CompletableFuture<ResponseType>`
Meccanismo di callback	Richiede un comando `AsyncHandler` con metodi `onSuccess` e `onError` separati	Usi `CompletableFuture` APIs forniti dal JDK, ad esempio`whenComplete()` , `thenCompose()` `thenApply()`
Gestione dell'eccezione	Utilizza il metodo `AsyncHandler#onError()`	Usi `CompletableFuture` APIs forniti dal JDK, ad esempio`exceptionally()` , o `handle()` `whenComplete()`
Annullamento	Supporto di base tramite `Future.cancel()`	L'annullamento di un genitore annulla `Completab`

Modifica	v1	v2
		`leFuture` automaticamente tutti i future a carico della catena

Completamento asincrono, gestione delle differenze

Modifica	v1	v2
Implementazione del gestore di risposte	`Future<ReceiveMessageResult> future = bufferedSqs.receiveMessageAsync(receiveRequest, new AsyncHandler<ReceiveMessageRequest, ReceiveMessageResult>() { @Override public void onSuccess(ReceiveMessageRequest request, ReceiveMessageResult result) { List<Message> messages = result.getMessages(); System.out.println("Received " + messages.size() + " messages"); for (Message message : messages) { System.out.println`	`CompletableFuture<ReceiveMessageResponse> completableFuture = sqsAsyncBatchManager .receiveMessage(ReceiveMessageRequest.builder() .queueUrl(queueUrl).build()) .whenComplete((receiveMessageResponse, throwable) -> { if (throwable != null) { System.err.println("Error receiving messages: " + throwable.getMessage()); throwable.printStackTrace(); } else { List<Message> messages = receiveMessageResponse.messages(); System.out.println`

Modifica	v1	v2

	("Message ID: " + message.getMessageId());
	
	System.out.println("Body: " + message.getBody());
	}
	}
	
	@Override
	public void onError(Exception e) {
	
	System.err.println("Error receiving messages: " + e.getMessage());
	
	e.printStackTrace();
	}
	}
	);
		("Received " + messages.size() + " messages");
		for (Message message : messages) {
		
		System.out.println("Message ID: " + message.messageId());
		
		System.out.println("Body: " + message.body());
		}
		}
		});

Parametri di configurazione chiave

Parametro	v1	v2
Dimensione massima batch	`maxBatchSize` (impostazione predefinita: 10 richieste per batch)	`maxBatchSize` (impostazione predefinita: 10 richieste per batch)
Tempo di attesa in batch	`maxBatchOpenMs` (impostazione predefinita 200 ms)	`sendRequestFrequency` (impostazione predefinita 200 ms)

Parametro	v1	v2
Timeout visibilità	`visibilityTimeoutSeconds` (-1 per impostazione predefinita della coda)	`receiveMessageVisibilityTimeout` (impostazione predefinita della coda)
Tempo di attesa minimo	`longPollWaitTimeoutSeconds` (20 secondi quando `longPoll` è vero)	`receiveMessageMinWaitDuration` (impostazione predefinita 50 ms)
Attributi di messaggio	Imposta utilizzando `ReceiveMessageRequest`	`receiveMessageAttributeNames` (nessuna per impostazione predefinita)
Attributi di sistema	Imposta utilizzando `ReceiveMessageRequest`	`receiveMessageSystemAttributeNames` (nessuna per impostazione predefinita)
Sondaggi lunghi	`longPoll`(l'impostazione predefinita è true)	Non supportata per evitare connessioni aperte in attesa dell'invio dei messaggi da parte del server
Tempo di attesa massimo per sondaggi lunghi	`longPollWaitTimeoutSeconds` (impostazione predefinita: 20 secondi)	Non supportata per evitare connessioni aperte in attesa dell'invio dei messaggi da parte del server
Numero massimo di batch di ricezione preimpostati archiviati sul lato client	`maxDoneReceiveBatches` (10 batch)	Non supportato perché è gestito internamente
Numero massimo di batch attivi in uscita elaborati contemporaneamente	`maxInflightOutboundBatches` (impostazione predefinita: 5 batch)	Non supportato perché è gestito internamente

Parametro	v1	v2
Numero massimo di batch di ricezione attivi elaborati contemporaneamente	`maxInflightReceiveBatches` (impostazione predefinita: 10 batch)	Non supportato perché è gestito internamente

Usa l'SDK per Java 1.x e 2.x side-by-side

È possibile utilizzare entrambe le versioni di AWS SDK per Java nei propri progetti.

Di seguito viene mostrato un esempio del `pom.xml` file per un progetto che utilizza Amazon S3 la versione 1.x e la versione DynamoDB 2.27.21.

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>2.27.21</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 per Java

Tutti gli artefatti Maven disponibili pubblicamente per il AWS SDK per 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	0x 07B386692DADD AC1
Tipo	RSA
Size	4096/4096
Creato	30-06-2016
Scade	2025-10-04
ID utente	AWS SDKs e strumenti < @amazon .com> aws-dr-tools
Impronta digitale chiave	FEB9 209F 2F2F 3F46 6484 1E55 0 7B38 692 PAPÀ AC1

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-----
Comment: Hostname:
Version: Hockeypuck 2.2

xsFNBFd1gAUBEACqbmmFbxdJgz1lD7wrlskQA1LLuSAC4p8ny9u/D2zLR8Ynk3Yz
mzJuQ+Kfjne2t+xTDex6MPJlMYpOviSWsX2psgvdmeyUpW9ap0lrThNYkc+W5fRc
buFehfbi9LSATZGJi8RG0sCCr5FsYVz0gEk85M2+PeM24cXhQIOZtQUjswX/pdk/
KduGtZASqNAYLKROmRODzUuaokLPo24pfm9bnr1RnRtwt5ktPAA5bM9ZZaGKriej
kT2lPffbBjp8F5AZvmGLtNm2Cmg4FKBvI04SQjy2jjrQ3wBzi5Lc9HTxDuHK/rtV
```

```
u6PewUe2WPlnxlXenhMZU1UK4YoSB9E9StQ2VxQiySLHSdxR7Ma4WgYdVLn9bOie
nj3QxLuQ1ZUKF79ES6JaM4tOz1gGcQeU1+UklgjFLuKwmzWRdEIFfxMyvH6qgKnd
U+DioH5mcUwhwffAAsuIJyAdMIEUYh7IfzJJXQf+fF+XfOCl6byOJFWrIGQkAzMu
CEvaCfwtHC2Lpzo33/WRFeMAuzzd0QJ4uz4xFFvaSOSZHMLHWI9YV/+Pea3X99Ms
0Nlek/LolAJh67MynHeVBOHKrq+fluorWepQivctzN6Y1NOkx5naTPGGaKWK7G2q
TbcY5SMnkIWfLFSougj0Fvmjczq8iZRwYxWA+i+LQvsR9WEXEiQffIWRoQARAQAB
zSxBV1MgU0RLcyBhbmQgVG9vbHMgPGF3cy1kci10b29sc0BhbWF6b24uY29tPsLB
lAQTAQoAPgIbAwULCQgHAwUVCgkICwUWAgMBAAIeAQIXgBYhBP65IJ8vLz9GZIQe
VawQezhmktrdBQJnABcIBQkRa8qDAAoJEKwQezhmktrdl1MQAIwEuDar3OTxkfTa
cPNKDnNzxaWqrxZ3FTQ+PyrHhQ6usxxrvDKJS+uCjE9bmWHVFU1R4yQNF+72lJdw
5UhX0u+ZgT9afApE65uAZuwLhPsz8upXT8C6VeKXh3shdw7qXi2hrwtM1a0Pls4O
Cs2C9rLUDMJTySrVDDVwpnaAB+8DcFrs9bIt5Q3gdOUatdzDvcb7QKh9jUvzCpbE
cInb1epDN5MRzowMR4iU2VV1RzLxCvm7CQSyXfgf0DFLkXWiknhOq9eINmytJFG/
ntFdiZFkNZ5hP709loywdfNrmqB6PsF8BPGFh8gKw1pjowrfHpv6cNIqShmA76LT
3OHViOlqGFB7obffq//eZGPR0oYJFDrOdD2CFRoHnP3N++AfkA4SRN7eXwyoz6Pk
Do9WNIEEkAcp6PGvv7AokogDo/4OqmxgC6fN+3BTOstWpv4FlD4NxOZWsTs49wxg
kPlCCVf8t75aZZkcjXng1eClZZQ5SBlRtSB7gMqtP7MIn2J5w8spNbs5xQvJc76u
NvzwEasPkY+UcHd05RddOUwoKqDerLUG7YqdlNCJoQR1mBIgZButbQlMyaZcmQqO
iR0kwDi9h6Dl6fnUb2dFCNJw+eDHvsjG8HI3IVZMl0bUQ2kmmwr1O2YQ1ynJQm0l
lMllI4hFU8/lHNHm8ie5darpVXQEwsGUBBMBCgA+AhsDBQsJCAcDBRUKCQgLBRYC
AwEAAh4BAheAFiEE/rkgny8vP0ZkhB5VrBB7OGaS2t0FAmUkSiIFCQ+P/Z0ACgkQ
rBB7OGaS2t3HVg//S+/KbeOBf+nCdHsrWtp9kxvWIpAGvQhIbxx1tp/impfm+5Rm
fKPD0KX+g42fuMmOdDE4gjO4GjGd7ZY3bx+0zbDSdVebzmYCbPZ/BDP990oPKidd
w6Gl8PaIyqfuARKOESBETvAwNgwO4t2ocjs4pYZV+CuHvESYpqkuHjmHtye6ajZW
Mv24NhjVo4EfP33dPugTjXLjeuGT7qQpsYV3a66juHmPVkXwuPqxh9wTNc5TU6FG
UPSfIGMPL0xha7Rg2i5zvRaAxx4bHqGO8IAz/l/E/tJkV5xnt494HQam9UDbiFI0
Q0TSve1R6S45/UjQW6cycyduHtk72s9ipa9YM0ilTdLgKMWFjzYv4h4qeYvLw3oB
JGQew+I0I4dIrwL/TKet33EuFfwmyT9MaJBhqV6geFaQ0uVmwvzpAcvxIoSqSpkJ
B/kASqCEM/o0QZLuWc56cDsmMisD0ouVPt+c1Zk7AWLlf6j8LKYTbK0QxLRh/eeZ
jhSf3HnpaCfonbOoHmeo5d/o3EZ0AiA4GbT3xgScoIgXOT7KGgOWmtWkdyd3Zvl/
o6q6Hwpg4RsqCkwnfNm5ZIvmTAYXWc2hiICSicxrP0fek5Cc4xVgJR5RMNGyI7+m
ut1SE2WvLhMCwEyl5ecWF0tUze8VBlWkHJp0Y4k2ado39Zq/DZrTRQYEVrjCwX0E
EwEKACcCGwMFCwkIBwMFFQoJCAsFFgIDAQACHgECF4AFAmEyoZoFCQueVRUACgkQ
rBB7OGaS2t3axA//dgcz5T4Fs7LWdIQB/KRnvX64IzaGQcwt3FBhYH+sFaSaD+lu
752vi3jlGxMBKs2NFxk6e2U8xBEir3vfKYd+mZ5L6egXC9MYfvMO/UFEFH3A+t3V
dTOJK4RQcaL9+rFRVdDmZuifN9OFfy25d66JCZ5OiqgHTQViVmbrGw2cQdyNWXrq
YLgg+gktadkmzis4J6hF/le8NOBfrG3n+QthFl/v2ppYYW9pmmxzUIf6tAlR1Vr8
PhPukjuFfrPLRL3XPiK4LkdlSI5MX7Llq4RkcZN1NY1LWS8699wJOLRcr8aQYvzZ
Jm7tfZUaekJ5ScXJWJEaWT4poMbWxXINj6VwE+DqKWvjkzoxBXSIdLk4XThA1dIq
3n5SfMkfuWV2A2xHqJtEzI1XeTm/d/JRxIG8hjIs5FNMGJUSNANJuTVA2putCVf0
JbP4QlafXoEVOYwW/EFJY+brjQadwc/knf/QxszDcKb3THuTxR80A0h6ZysmtLEg
PCaD1xUCDdr4P7DGOtV7yMaDR6O8QKmb7TKzCKCPnHouqPT0DhSB2MRq437+mfSe
EGga/sPMxe0z8ugO2CnrCf3eplU3ZOy4eeQSThTKe0Hp5YlsF1cdZSvjt7GJaHR2
9A22nAJY9PojtlM+0Dkr/PH6r2brv3sEuACRNhzqCWhAve+zVnVLeb+Fk1rCwX0E
```

```
EwEKACcCGwMFCwkIBwMFFQoJCAsFFgIDAQACHgECF4AFAmCavPYFCQsGcHEACgkQ
rBB7OGaS2t3fqg//asHYSTBI4IoLibSF5ZJ8NaLo4EqgRtsO0W1zr8fCoFbxI+yI
qWrNXR7eoFj8tWO7Tj6S0kQr4QZcucLeILfox6CtDzO3f3WQTH9m/0si5U4Jf3RA
gBd0vwxVSSSNEsIUUfylOBHLVwlhaTVfh1h1dZhzb18LA1Vqu0100GGxvaG3dUl4
oMSRSK8EeaSO4hIM8ScjVyhjsuPRm1jlwd1EXZ5mkHRGRMGmwi3XysJjIeQRFmuG
a9uPes7I/K85r7X91BLz+G/mkSZsrHxzLxF0mSZtQdhq08GyMeQlJavs7sb1UVbg
ag30JEAjgqRsNIQOMIvl2/amrRJ7DUyQu5YkZQsAyCIhSVZw6zlJlkYVKSw1Cu1I
VX1n/Aahx5wwaQZA1dDTolX8WvL9O/KmEGWbKUSov40uoRwSOeXP3QhW75kD4zT0
n3pSUc7tXka8GAz5Ar+r90l4wc9as2WjWADo3CX8cF0zQOrN1naMQ++xBIAG9ms8
y3L4ECoRqvjCpjUAfhflxwSM7uclCgFBINFKSLV+QGxzWfHjg3+bSQd0hrRn9j4z
Di9aJmFESQ6iykbUjiUFrcI1NUCFKz2awaxh+Sq8UkYcvuxljxdK/zYYEG8DSdQ2
uwlssHCjYVpAb16hq1EAaSqnhv86020G4Z6JCg3AUZt9RlMBpK2Pn/NmPSzCwX0E
EwEKACcCGwMFCwkIBwMFFQoJCAsFFgIDAQACHgECF4AFAl771vAFCQlnimUACgkQ
rBB7OGaS2t35Jw/9GhpQquJVUCAsc4az7+ad8q2d9OUKXOLl2x0j3/ofS8umZJYr
8KXZxPqBqvLjlUyAB5Ur4fWVBdp9iP4Vj74jmRih7YatK8heGmNoUAnI1/90qV5O
ypFObfvWLxvcUNizmS/LYKdNuYcHwyw4bFyTz9gd3XH6Dxak7YiAXz3JgJIXcIB3
ELfpsduzpncLeVwz/dKhYX5iJ7Y/xd4Ew8Ian8FyNDNRwcXobTpPAMNiGLG5xSLq
8RgQfkm07BVVDtu5tPw4V46gnBXGXNjPhSirGMoNbKZqtPO7TcBQhPQ9c95x73hs
kAqZKHrwi1B7cfy1I8y5sRFbPa0RXeVWQKEMZdwLsFhCbEex6iXHP+rMK0nSJf0G
9lF2eJk140n8K7wzakOnjvN00VfN/9D+xD2lCXCzbYbaDXX2tYOd4GS72fW9M/zT
96tIH5UtMGMOICuUJjnL34EbzBuxwTEnJKhDGQH2P+eUzM9jmuCTRLLGw2P0Zuof
6GsSNLf+5pw3BIvEZw5PYT1N6i3ml9MQ7p9BWr7flCF8CU/xzyPN7TVb/677Be7V
SL1rNFLNi0IdqcbLJ63pTWaUGkCSqRnMfq3cIDlzNz2LoVvOO8VVmdtFRkhHN8hJ
h7CUXlaqBOfaJhV3FqA9G8sXmSN3r3pusZb13kfCKsJUZorThWxdaUBuUH3CwX0E
EwEKACcFAld1gAUCGwMFCQeGH4AFCwkIBwMFFQoJCAsFFgIDAQACHgECF4AACgkQ
rBB7OGaS2t1aKA//dBaXto7zUFRbJvLgcSE3hJ0ChYZj8Uuo7iYK26mVycyBFQJK
Iv2XYFIJMwK1Rx1Ja1jA6BIKE3aYyx2LVTYU1Hke826dhH+kV2qN9C6t/VmYpV4b
n/i64po7EzD17ykrm5gB+z47uCvyC79/r8Ouns1mTf/JUql/hYJj2hf1MDWr07/X
Wc1zBLjlT93tg/43Vgz1wFdrUOcNx1+bQGXKTOi4AXSp3UvCL+2YsQ2/JsPf7ZzZ
awSuUVkZJrllp87S1MhZeHmTrCHkV1FHJyudLJErcH0337Jpd8xDRek7Klrx2pAS
cKIiyeAMIk/G1OuExYqEEVFQzMr9ZlMNuhNBJAMr5hVGsTgrwy2hlIrBktXav07d
OleSlsqw9ViZ7F6t90x5l+NkwXVsLsGYSzuNyIfomNuoCgA+cfM3TjzVp41qsg1J
WJc9Bavaan2pKF6Lb9Fq8u3HZk2u+YZbvZkqkXwTdZZQOkEmoVV4Y1G86bdpmPyj
8eV7C02NxPii4l+qV8qJQu/6DsA0QwMtBMUNODm3BF2+ZmUHuhMGxq9/4vDE8heE
ffYhHtNftV6JwwzGZmeZkrYA1P9AGLeVp/6iNUe8H5/oPvh4s2rRmqN+L/dQU1ix
iOAT5iAKoRQGkduXrWc4fAY5KxDB9qna4oqXO6QP8rEflI8ELcNgYEj4oJvOwU0E
V3WABQEQALzM0Cs9Zvd08xOEvbEBj59LrS9d0HVKQ61gmkNakWC+jR35VD6FXpe6
UYAcBLrEbVYfKw9P0p6MhFKAsb570JoznKGzE1rVYUZQzhD0RKje35rvkajvEcjG
AWMLTjr87pWHeD0389ER64bzORncfa/l+YP56PI+CThb2wUvTTONGJkPQUpVhH+P
256cQL/Y0Fwu4XLerpwN+YKgMQ47raRcydobPeSfMQr9fVKRyOzFEOrvNpCVDUqi
77d0gLDLjHlIlDyOX5554S8XYLb91eYOiFvnu2pTCKiiExRCSYK29mAQePKlTCCn
QxOjbmBbGS8mVIkpQ5vpvXvzpY3JIjMXaDGqWSQSYGXhECyxCR5eOtKYbCwwPIc2
rIl5gW6yXyw9pKmj5XafTP7YHTvRSr7CZ/VLkDkWl6AfQ9nPOg1mjwjpDFpmN71h
JlSKMaZkh0QGV5FW3dK+GLwxiWdqx3htbZErWyvumWQF/xBF7puKJBEXcoM5KfkJ
```

```
uZekBwcnVkfNFF2RdkM1ALq8InGzLXc7ROuEm0BXVirfju7JRtWLb3UhJWCuhRW2
muyYegSTkag5MduD1IJK37GL8WIlAL65taYgZegUoxHdSaEOefOhspxuduz8d33z
UV1WCFhi+r/+BMCQmTRbF8ao7fTC1dGd084DRP6qE/dMT4u0ZEn7ABEBAAHCwXwE
GAEKACYCGwwWIQT+uSCfLy8/RmSEHlWsEHs4ZpLa3QUCZwAXCwUJEWvKhQAKCRCs
EHs4ZpLa3XtzD/9dwi1qffV70UTq8w/21jn1owHp09jxP7WHTmPWHE0BW5yFIWlV
A1gKN6Ym0dw+LvS5WOKJaRnyewUyBxWvZsn6Wlb5qzY7nmCOKJpYtuCUPwiqjXWP
EM8c/v0MojSuwMOXBAViLvOFhgdUrHn1lk962XvWAW++4DXFh2deaV0163IFMRmO
PNPDAiPWBVqvBANIh2sLRZ5gd1BXwpVrd+x8tzyr69YrN7hutPlCyPEUM9//mcEh
vFPsbW/iOx/foCE3NXhQm/rSMKecVn5csXBV2JOlMzi+8txYNrSBLkjbSB1AvTQ1
aG3+nCNCgM2XDLyoj0IrgZ1To4Ay5gmTOR+msY/cfoIuKFYenmtxy6jM8o5uSZHg
hoClrx9IA98hhGQ73G2r5EDpXuU/uCXn53Sswj65bl9IssfqEIoji/FonkkpEgeg
bGXFDUnrhicDO/WOzqpXf2Fa0DQWY+Vc/pt52ftBFgwzCNIUYDKUhCHPnZ0wtLtd
N2fkXHNiCavCDZlOud7FHHwmRNdj2q1uKxe4m+pFYmKwAU/H+Htkz9Gjsj+ZKedY
nnfai2s2gQOrbfwvV9VdhCWSuLK17ZnGTtiJuOUQIlV8n6QQJpohd3mVgmynu6gQ
uKw0YS2RuEUFv0vOg2tASA+4EM/SBUpGhudODLA4b5wO4gKmh1B1HqQrIsLBfAQY
AQoAJgIbDBYhBP65IJ8vLz9GZIQeVawQezhmktrdBQJlJEokBQkPj/2fAAoJEKwQ
ezhmktrdwMAP/RpFylIL4yhgscBOEnQ7e3No8OraNk0z/YhSd125N/uQVEU94JGQ
rrvQ+4Lfve2laPweBDO18/A0CsmOyHPVQMA0a2vx8ItVdIcNc8iFkP4AJ1922lOq
i0Vh0b1UeZnlfK9+Qvq4PQ2lhWJr0uzyL/S38REsAT1I25sfJOP+RCaR1MH9dm85
E56Lee6uZR8SkGuiL6kGpPh6fWTNij3bICjth1iSSCL2HCOW8lvcwSldDu2EfILU
QCSqfSG7bF8dFk+nKhzhVXOUks3XGjLdICxZewU5ycryitpfRgARgZs2A43gshdi
fiKaX6Ksan03uhKDrLhDHNj2y07PUrFo8ggtlRpV/PrlB/UqCsC9FUOixbD+n4ZF
Sqov2qwelLj0f4mZ6yiLsTDUOFPrdkOlHTJZl7AF0zXZMM6CvaCUaJCKx9GVdSrR
+LI4wLQonPrTnXavhkC4intlqSX8ZQNLhEggdE8YwMEJn59R/nVIT3i5WzYph5R9
P4Vz3Yn7jRqM8wAyEbHkA8s45fMRi9akWSw93H5nWukcmfkt3UEbmka3BQg3HKWP
6TvhfI28euM8qqjbPilfkpEBjnChYVk2Rgn0P8zA7Q5kCo293kwJL9c3RDjMPcxI
45ktKvBTZftsDt1Z718LwW7Q3VQiGiKvo1XLMuV7Z51fmydfUPcrnv17wsFlBBgB
CgAPAhsMBQJhMqGaBQkLnlUVAAoJEKwQezhmktrdbhwQAITmFb67XIUZswr3TREd
Q7ZCLG4EDyfTsW8n75r6A9OqsR+z68nC2Sm7e8mKQFFPwjHPOhsGhHtCOTZtQk7O
jbwyL4N3uxDyEvOfbckH5WzOejZcG7KKQrqAiWJJ7q6CH/zOnVurySjVyzJpy/wL
WpVAcF/uaW5ZhlFCXqePaEzsUBJ757qsr2ho14BV4seT1RSQ9nneTZ0Hhab3wqXP
4qDTo8+zkTvNo9YbeZ1qj62l1+QGIUBTP5MEdXCuC1e4FQ3f6vnXxmB86cUPx7cl
/y2rIjeiOdkKgPeUjNWWSzxS2jYehL5we7gvaSwmEvJ74pV+/3Hs+TxX39XtYFwj
k9I795idnsS5l1dAW3yoI3HBQsYa3US7bpH4g3yZMkstc3bHJ6X54PMCd8Skb+N3
FE8+zGduDmDTKitumiWVVxEFGIwsLAcpWPxecI2AMIMGfMheURYsdvD/yvCbCB29
0KwCSrDVkAG9N2VorNzd7KUeTPTMN1bg2d1lF6u5sQeTN5KVaGd7xE1OXME2wA2D
T3+EsAQytriFbcWm3s8Ugbc9BXMmKBfjlvKu6+Fr6Mgvf/txn56M2SyXBCFQ5OFt
qTFuAFIRv+nayk5tx5Eg1iA7u3dbB1jH3yxGH1B7TeQypA5BqD3x72b7vbXkeci3
1Kz035LYoT5/yTK5sGvacIvCwsFlBBgBCgAPAhsMBQJgmrz3BQkLBnByAAoJEKwQ
ezhmktrdLmgP/1FkWkYhxACnkagRv09mpPl2STbu0B3zYKFBALm/Wa7vKDz18dgC
S3BxDSlpnhZS8QA3VjmbOAZvaDnsN1UJ0f3Qao5136G/UXPnmFIwN612szP0K6nF
PEsotzIzRlJo3S+WkBfiKaQDIDgSxtUxJzOwufz76xibmKRhJ5ChMDCvxmIaoNle
tKRxFT77OupnnyaQs22UsueqrZJ0resgTVnNeF4A1+lU59pFuAlf971SVLr472LP
Uj8mPJihF2ukL1Hdz3F7+kYlpOJRmLk9fo4dlZHBUPiZ1ML/U2yhQfW+Y6tW71vf
```

```
izAxJWF7se6QT+UT5Pji6cohMSERVoYt8e2jFjs0PiPcrjU3mJEx4hAEEVIbP9RY
eKC4CL/UGYAtJkUjd85vKZUHYr7NWZQKLAKqpAPQUMKrIKLEHuz/doq2CCamstLI
vcBgg8EjLJnl3SBesFt/1DCWZeummqz3omQKRl9EHU2cIzIf0Cv/IEysnmbpSpjZ
DX8Fqjtezoq1qiyrLFR7YN1VDPBCHYfqDagw10nlrWFJqT6VqfslmdMdTBRWYVEB
0GUxrkyI+APdi0M2634/410b1ptkqyTIr1KIg1J/qsSiKVcBZSOYFW/rskxYcPPT
wpKYaycEYtOdkS6FPcnehJOOlB+F32WVq2bs2Ps8we6KhjjaYS4Iv4dwwsFlBBgB
CgAPAhsMBQJe+9W/BQkJZ4k1AAoJEKwQezhmktrdvzMQAI6BBj3c2r4bDpV3TwkX
dQ+UCa/E/zUhFds9XKfGb3a5IzRdPUwT+KrAZyiYrr2NSMOzhl/VtqJL18YCYsxO
Ob/TB1hDM+IZiI5gH0cHKhDYKTnNSGP09P/pJAlvHQend9CdZE9J9jwkczfS+bz6
mVxkxpi73fTDox9duesOLsS2/ntRzA0wqhDdaaavRvhAEf9vavCWVrNZmq22WVsU
lnIPxNWGGzWn85JYI6uAi4f4/ABFkry69/c0cvbr0P8qgCmeCuGmX4f0j7qRg77A
+mSueBDx8RK0O2ol021B7b8IcVizj+lpsRQN0oa+i+mFG+o6vtD1ZYhQude4N5sR
RybcLclxjSCoZs5q9JfTpbB2n7pSf/UD3ytwnt9kpD4Vv9dTGAPB83bjL+QK6e3A
XM10jxFE5jSFSr94E40kK80YcIR5jLqsg2f610ENY5drMSA4zuDFDL1Y2ChfjgjZ
uNoFbPHGt/8DfWTVOochVnikA7ggKjz20+RjvwyrHhRMAftO8MMh9UV28pdL+H53
oOtOV0u5aoTbcNqdYQy9B2Bw4lfmj2fi6Dpl+vnZp6hOm0CWiJVW/dtilppYjuxd
w5Kj+9IxZYaBNYH4l1pMT+BsvMDqGzXxDIL89NnY5BkMvqEKnjXSHGRWYMz0xigf
51YKbfQnEQ1oz5bRQndntRQWwsFlBBgBCgAPBQJXdYAFAhsMBQkHhh+AAAoJEKwQ
ezhmktrdTyEP/0HOVWHwQsaWjMrGj0OOMFzxGUo8SBmYYTBs29VM8wBGDsPkYCje
ZzU16i9iqDpDqxpyqmTigcjHV8CDx/6xsMBLG2yKaKZ4m3+YnOQf/sQkyCvqiyMF
9mS7pDYWy+mPhPuw8TDIfiqgVhzjSpIMFWPqxVjn6KKbPN/QASr3Pf0cuP6qpHG+
NAM6Q5dYkCebyvwzLmg1sVnil6iSyJd1jBj3D34XrgWS9buyxBB2CjIM76WxfNVi
J9zAaPI78X9v6PpDGn0kg6oLzrusrvBjoZknKQm0SZ+41fx6xvrTPs8uPEzevzJB
lkke6kw9+KagY8mrVX1ZenRg+sY/4vxJreYWQeql67ggx+wFjKDcfhZA7m7OLHOD
ysrGVCLcmuinUBaNlHmLDcGYXZ+kMCoXfObpuCVByQmNJgEb47EIFlx/+TEeNHKM
O+22xL1atFzXfkEVZck+NghLZyFDhS3g1bma7puU7r752uiJjA6Iv8+kHDXi+/V7
GNpuiEFUYh69QQ2//CS5H51osC/Bkb9evSn/Lp8dMubtWAaXDGJMgw9vqZ55N02N
K0fvF/IKHnGkvH28rv0OPCvOWTA/MClv28y0PrSvcvMXnduLtkBEX7TISMPW+n+O
Ta63/z4YFfEZ7sFLrEm3Q3vJMN3mE5i3cw+JGXPSu0nTtgqk/oZv//SS
=bboB
-----END PGP PUBLIC KEY BLOCK-----
```

Chiavi storiche

> **Important**
>
> Le nuove chiavi vengono create prima della scadenza di quelle precedenti. Di conseguenza, in un dato momento, può essere valida più di una chiave. Le chiavi vengono utilizzate per firmare gli artefatti a partire dal giorno in cui vengono create, quindi utilizzate la chiave emessa più di recente quando le validità delle chiavi si sovrappongono.

La tabella seguente mostra le informazioni chiave di OpenPGP precedenti per le versioni di SDK for Java 1x e SDK for Java 2.x.

ID chiave	AC10x 07B386692DADD
Tipo	RSA
Size	4096/4096
Creato	30-06-2016
Scade	2024-10-08
ID utente	AWS SDKs e strumenti < @amazon .com> aws-dr-tools
Impronta digitale chiave	FEB9 209F 2F2F 3F46 6484 1E55 0 7B38 692 PAPÀ AC1

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-----

xsFNBFd1gAUBEACqbmmFbxdJgz1lD7wrlskQA1LLuSAC4p8ny9u/D2zLR8Ynk3Yz
mzJuQ+Kfjne2t+xTDex6MPJlMYpOviSWsX2psgvdmeyUpW9ap0lrThNYkc+W5fRc
buFehfbi9LSATZGJi8RG0sCCr5FsYVz0gEk85M2+PeM24cXhQIOZtQUjswX/pdk/
KduGtZASqNAYLKROmRODzUuaokLPo24pfm9bnr1RnRtwt5ktPAA5bM9ZZaGKriej
kT2lPffbBjp8F5AZvmGLtNm2Cmg4FKBvI04SQjy2jjrQ3wBzi5Lc9HTxDuHK/rtV
u6PewUe2WPlnxlXenhMZU1UK4YoSB9E9StQ2VxQiySLHSdxR7Ma4WgYdVLn9bOie
nj3QxLuQ1ZUKF79ES6JaM4tOz1gGcQeU1+UklgjFLuKwmzWRdEIFfxMyvH6qgKnd
U+DioH5mcUwhwffAAsuIJyAdMIEUYh7IfzJJXQf+fF+XfOCl6byOJFWrIGQkAzMu
CEvaCfwtHC2Lpzo33/WRFeMAuzzd0QJ4uz4xFFvaSOSZHMLHWI9YV/+Pea3X99Ms
0Nlek/LolAJh67MynHeVBOHKrq+fluorWepQivctzN6Y1NOkx5naTPGGaKWK7G2q
TbcY5SMnkIWfLFSougj0Fvmjczq8iZRwYxWA+i+LQvsR9WEXEiQffIWRoQARAQAB
zsFNBFd1gAUBEAC8zNArPWb3dPMThL2xAY+fS60vXdB1SkOtYJpDWpFgvo0d+VQ+
hV6XulGAHAS6xG1WHysPT9KejIRSgLG+e9CaM5yhsxNa1WFGUM4Q9ESo3t+a75Go
7xHIxgFjC046/O6Vh3g9N/PREeuG8zkZ3H2v5fmD+ejyPgk4W9sFL00zjRiZD0FK
VYR/j9uenEC/2NBcLuFy3q6cDfmCoDEOO62kXMnaGz3knzEK/X1SkcjsxRDq7zaQ
lQ1Kou+3dICwy4x5SJQ8jl+eeeEvF2C2/dXmDohb57tqUwioohMUQkmCtvZgEHjy
pUwgp0MTo25gWxkvJlSJKUOb6b1786WNySIzF2gxqlkkEmBl4RAssQkeXjrSmGws
```

```
MDyHNqyJeYFusl8sPaSpo+V2n0z+2B070Uq+wmf1S5A5FpegH0PZzzoNZo8I6Qxa
Zje9YSZUijGmZIdEBleRVt3Svhi8MYlnasd4bW2RK1sr7plkBf8QRe6biiQRF3KD
OSn5CbmXpAcHJ1ZHzRRdkXZDNQC6vCJxsy13O0TrhJtAV1Yq347uyUbVi291ISVg
roUVtprsmHoEk5GoOTHbg9SCSt+xi/FiJQC+ubWmIGXoFKMR3UmhDnnzobKcbnbs
/Hd981FdVghYYvq//gTAkJk0WxfGqO30wtXRndPOA0T+qhP3TE+LtGRJ+wARAQAB
wsFlBBgBCgAPBQJXdYAFAhsMBQkHhh+AAAoJEKwQezhmktrdTyEP/0HOVWHwQsaW
jMrGj0OOMFzxGUo8SBmYYTBs29VM8wBGDsPkYCjeZzU16i9iqDpDqxpyqmTigcjH
V8CDx/6xsMBLG2yKaKZ4m3+YnOQf/sQkyCvqiyMF9mS7pDYWy+mPhPuw8TDIfiqg
VhzjSpIMFWPqxVjn6KKbPN/QASr3Pf0cuP6qpHG+NAM6Q5dYkCebyvwzLmg1sVni
l6iSyJd1jBj3D34XrgWS9buyxBB2CjIM76WxfNViJ9zAaPI78X9v6PpDGn0kg6oL
zrusrvBjoZknKQm0SZ+41fx6xvrTPs8uPEzevzJBlkke6kw9+KagY8mrVX1ZenRg
+sY/4vxJreYWQeql67ggx+wFjKDcfhZA7m7OLHODysrGVCLcmuinUBaNlHmLDcGY
XZ+kMCoXfObpuCVByQmNJgEb47EIFlx/+TEeNHKMO+22xL1atFzXfkEVZck+NghL
ZyFDhS3g1bma7puU7r752uiJjA6Iv8+kHDXi+/V7GNpuiEFUYh69QQ2//CS5H51o
sC/Bkb9evSn/Lp8dMubtWAaXDGJMgw9vqZ55N02NK0fvF/IKHnGkvH28rv0OPCvO
WTA/MClv28y0PrSvcvMXnduLtkBEX7TISMPW+n+OTa63/z4YFfEZ7sFLrEm3Q3vJ
MN3mE5i3cw+JGXPSu0nTtgqk/oZv//SS
=Z9u3
-----END PGP PUBLIC KEY BLOCK-----
```

Cronologia dei documenti

Questo argomento descrive le modifiche importanti apportate alla AWS SDK per Java Developer Guide nel corso della sua storia.

Modifica	Descrizione	Data
???	Sono state aggiunte informazioni sull'utilizzo di `LegacyMd5Plugin` per la compatibilità con le versioni precedenti con i sistemi che richiedono MD5 checksum.	19 maggio 2025
the section called “Differenze tra i client S3”	Aggiungi le differenze del client S3 tra la v1 e la v2 di AWS SDK per Java e fornisci esempi di migrazione se lo strumento di migrazione non è in grado di migrare automaticamente il codice V1.	24 aprile 2025
Differenze di deserializzazione	Aggiungi la differenza di deserializzazione tra v1 e v2 dell'SDK for Java.	10 aprile 2025
the section called “Suddivisione automatica delle richieste SQS”	Aggiungi contenuti di migrazione per il batching automatico delle richieste SQS dalla v1 alla v2 dell'SDK for Java.	8 aprile 2025
???	Aggiorna le informazioni sui calcoli automatici dei checksum	3 aprile 2025

Modifica	Descrizione	Data
the section called “Configurare ALPN”	Mostra la configurazione della negoziazione del protocollo ALPN con il client HTTP basato su Netty.	21 febbraio 2025
the section called “AWS Lambda funzioni”	Aggiungi informazioni sull'utilizzo di EMF Logging Metric Publisher con AWS Lambda per acquisire le metriche SDK.	6 febbraio 2025
Implementare `ContentStreamProvider`	Aggiungi un argomento su come implementare un`ContentStreamProvider` .	29 gennaio 2025
the section called “Protezione dell'integrità dei dati con checksum”	Contenuto aggiornato con dettagli sul calcolo automatico del checksum.	16 gennaio 2025
the section called “Lavorare con Amazon S3”	Aggiungi contenuti di migrazione per lavorare con Amazon S3.	8 gennaio 2025
the section called “Accedi AWS ai client HTTP basati su CRT”	Aggiungi informazioni su come utilizzare un jar specifico della piattaforma con componenti basati su CRT. AWS	14 novembre 2024
the section called “Usa IAM Roles Anywhere per l'autenticazione”	Aggiungi informazioni su come utilizzare IAM Roles Anywhere per l'autenticazione.	6 novembre 2024

Modifica	Descrizione	Data
the section called “Configurare un provider di credenziali”	Aggiungi un esempio che configura un provider di credenziali utilizzando la `asyncCredentialUpdateEnabled` proprietà.	4 novembre 2024
the section called “Utilizza il raggruppamento automatico delle richieste”	Aggiungi un nuovo argomento che documenti l'API Automatic Request Batching per Amazon SQS.	23 ottobre 2024
Chiave OpenPGP	Aggiorna le informazioni chiave correnti di OpenPGP.	10 ottobre 2024
the section called “Usa tipi complessi nelle espressioni” e the section called “Aggiorna gli elementi che contengono tipi complessi”	Aggiungi contenuti su come lavorare con tipi complessi nelle espressioni e negli aggiornamenti.	10 ottobre 2024
Aggiorna i nomi dei bucket Amazon S3	Aggiorna i nomi dei bucket Amazon S3.	30 settembre 2024
the section called “Utilizza endpoint basati sull' AWS account”	Aggiungi informazioni sugli endpoint AWS basati su account per DynamoDB.	24 settembre 2024
the section called “Lavora con attributi che sono bean, mappe, elenchi e set”	Sezione di aggiornamento per DynamoDB Enhance Client che illustra come lavorare con attributi che sono tipi complessi.	6 settembre 2024

Modifica	Descrizione	Data
the section called “Configura i client di servizio per abbreviare le ricerche”	Chiarire l'uso di `EnvironmentVariableCredentialsProvider` quando viene utilizzata SnapStart Lambda per Java.	19 agosto 2024
the section called “Configurare il supporto per il trasferimento parallelo”	Aggiungi pagina con informazioni su come abilitare il supporto per il trasferimento parallelo	15 agosto 2024
the section called “AutoGeneratedUuidExtension”	Aggiungi informazioni sul DynamoDB Enhanced Client `AutoGeneratedUuidExtension`	14 agosto 2024
???	Aggiungi una sezione sullo strumento di migrazione (versione di anteprima)	9 agosto 2024
the section called “Notifiche di eventi di Amazon S3”	Aggiungi una sezione che spiega come lavorare con l'API S3 Event Notifications	21 luglio 2024
the section called “Mappatura/documento DynamoDB APIs”	Aggiungi informazioni sulla migrazione da v1 a v2 per la mappatura/documento DynamoDB APIs	21 luglio 2024
the section called “Notifiche di eventi di Amazon S3”	Aggiungi le informazioni sulla migrazione da v1 a b2 per l'API S3 Event Notifications	21 luglio 2024
the section called “Tentativi”	Aggiungi un argomento sulla strategia di riprova	18 giugno 2024

Modifica	Descrizione	Data
Come impostare il TTL JVM	Rimuovi le istruzioni per impostare `networkaddress.cache.ttl` la proprietà di sicurezza utilizzando una proprietà di sistema della riga di comando java.	21 maggio 2024
the section called “Riduci i tempi di avvio dell'SDK per AWS Lambda”	Aggiorna i consigli del client HTTP per ridurre i tempi di avvio per AWS Lambda	14 maggio 2024
the section called “Metriche dei client di assistenza”	Riorganizza gli elementi della tabella delle metriche	1º maggio 2024
the section called “Risoluzione dei problemi”	Aggiungi un argomento per la risoluzione dei problemi.	26 aprile 2024
the section called “Metriche raccolte con ogni richiesta”	Aggiungi nuove metriche riportate dall'SDK.	26 aprile 2024
the section called “Imposta il TTL JVM per le ricerche dei nomi DNS”	Modifica il TTL di ricerca DNS consigliato su 5 secondi.	23 aprile 2024
the section called “Dal nome del pacchetto alle mappature ArtifactID”	Aggiungi il nome del pacchetto all'argomento di mappatura Maven ArtifactID.	17 aprile 2024
the section called “Pubblica le metriche SDK”	Aggiungi i dettagli di configurazione alla sezione delle metriche.	12 aprile 2024
the section called “API IAM Policy Builder”	Aggiungi le informazioni sulla migrazione dell'API IAM Policy Builder.	11 aprile 2024

Modifica	Descrizione	Data
???	Aggiorna le informazioni sul proxy HTTP.	3 aprile 2024
the section called “In modo sicuro”	Aggiungi istruzioni per disabilitarlo IMDSv1.	14 marzo 2024
the section called “Step-by-step istruzioni”	Aggiungi istruzioni per step-by-step la migrazione.	8 marzo 2024
Migrare alla versione 2	Aggiorna l'argomento sulla migrazione.	14 febbraio 2024
the section called “Configurazione AWS dei client HTTP basati su CRT”	Aggiungi informazioni sul client HTTP sincrono basato su AWS CRT.	5 gennaio 2024
the section called “Amazon Cognito Identity” e the section called “Provider di identità Amazon Cognito”	Gli esempi di Amazon Cognito sono stati spostati nella sezione Code Examples.	28 dicembre 2023
Usa le funzionalità dell'SDK	L'argomento relativo alle funzionalità dell'SDK è stato rielaborato.	11 dicembre 2023
Chiave OpenPGP	Fornisci la chiave OpenPGP corrente.	6 dicembre 2023
the section called “Differenze di serializzazione”	Descrivi le differenze di serializzazione tra v1 e v2 dell'SDK for Java.	5 dicembre 2023
the section called “S3 Transfer Manager”	Aggiungi una sezione che descrive in dettaglio le modifiche in S3 Transfer Manager dalla versione 1 alla versione 2.	13 novembre 2023

Modifica	Descrizione	Data
the section called “Riferimento all'annotazione”	Aggiungi un elenco di annotazioni di classi di dati che possono essere utilizzate con il DynamoDB Enhanced Client.	30 ottobre 2023
???	Aggiungi informazioni sullo stato della migrazione di librerie e utilità da SDK for Java v1.x a v2.x	17 ottobre 2023
???	Aggiorna l'argomento sulla configurazione di Gradle	17 ottobre 2023
the section called “Ignora gli attributi nulli degli oggetti nidificati”	Aggiungi informazioni sull'annotazione DynamoDB Enhanced Client. `@DynamoDbIgnoreNulls`	22 settembre 2023
the section called “Accesso interregionale”	Aggiungi informazioni sull'accesso interregionale ai bucket Amazon S3.	31 agosto 2023
the section called “Conserva oggetti vuoti”	Aggiungi una sezione che descrive l'annotazione. `@DynamoDbPreserveEmptyObject`	25 agosto 2023
???	Aggiorna la sezione client del servizio.	15 agosto 2023

Modifica	Descrizione	Data
the section called “Consigli per i clienti”	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
the section called “Creare policy IAM”	Aggiungi la sezione API IAM Policy Builder	31 luglio 2023
the section called “Inizia a usare”	Correggi diversi frammenti nella sezione Guida introduttiva dell'argomento DynamoDB Enhanced Client.	24 luglio 2023
the section called “Configurazione dei proxy HTTP”	Aggiungi informazioni ed esempi di supporto per i proxy HTTP per ogni client HTTP.	2 giugno 2023
Riorganizza il sommario	Promuovi le Esempi di codice sezioni e Lavora con Servizi AWS le voci del sommario di primo livello.	24 maggio 2023
the section called “Aggiungi dipendenza dalla registrazione”	Mostra le dipendenze di Gradle nella sezione di registrazione.	23 maggio 2023
the section called “Lavora con risultati impaginati”	Aggiorna l'argomento dell'impaginazione.	18 maggio 2023
the section called “Configura un progetto Gradle”	Aggiorna la configurazione del progetto Gradle.	3 maggio 2023
API client avanzata per DynamoDB	È stato rilasciato l'argomento Rewritten DynamoDB Enhanced Client API.	28 aprile 2023

Modifica	Descrizione	Data
Aggiorna le istruzioni introduttive del tutorial	Archetipo Maven modificato per includere l'opzione per CredentialsProvider; istruzioni modificate di conseguenza.	11 aprile 2023
the section called “Consigli per i clienti”	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 Best practice per la sicurezza in IAM.	14 marzo 2023
the section called “Ricarica le credenziali del profilo”	Aggiungi una sezione sul ricaricamento delle credenziali del profilo.	9 febbraio 2023
the section called “Configurazione AWS dei client HTTP basati su CRT”	Argomento di aggiornamento per la versione GA.	8 febbraio 2023
the section called “Lavora con i metadati delle EC2 istanze Amazon”	Aggiungi un esempio guidato per il client Java SDK per il servizio di metadati dell'istanza Amazon S3.	1 febbraio 2023
the section called “Usa un client S3 performante”	Aggiungi una sezione per il client S3 basato su AWS CRT.	19 dicembre 2022
the section called “Trasferisci file e directory”	Aggiorna gli esempi di Amazon S3 Transfer Manager per la versione GA.	19 dicembre 2022
the section called “Best practice”	È stata aggiunta la sezione sulle migliori pratiche.	18 novembre 2022

Modifica	Descrizione	Data
the section called “Carica credenziali temporanee da un processo esterno”	È stata aggiunta una sezione sul caricamento delle credenziali da un processo esterno.	15 novembre 2022
the section called “Metriche dei client di assistenza”	Elenco delle metriche aggiornato con i requisiti di utilizzo del client HTTP.	9 novembre 2022
the section called “Trasferisci file e directory”	Codice di esempio corretto.	2 novembre 2022
the section called “Riduci i tempi di avvio dell'SDK per AWS Lambda”	Sezione aggiornata con opzioni aggiuntive per ridurre i tempi di avvio di Lambda.	1 novembre 2022
the section called “Client HTTP”	Sono state aggiunte informazi oni di configurazione per coprire tutti i client HTTP nell'SDK.	26 ottobre 2022
the section called “Registra zione”	Argomento di registrazione aggiornato per includere i dettagli della registrazione via cavo per tutti i client HTTP.	4 ottobre 2022
the section called “AWS servizi di database”	È stata aggiunta una sezione panoramica dei servizi di AWS database e dell'SDK for Java 2.x.	13 settembre 2022
EC2-La rete classica viene ritirata	EC2-Classic andrà in pensione il 15 agosto 2022.	28 luglio 2022
the section called “Opzioni di autenticazione aggiuntive”	Aggiornamento alla dipendenz a richiesto per l'autenticazione Single Sign-On.	18 luglio 2022

Modifica	Descrizione	Data
the section called “Transport Layer Security (TLS)”	Aggiorna le informazioni di sicurezza TLS.	8 aprile 2022
the section called “Opzioni di autenticazione aggiuntive”	Sono state aggiunte ulteriori informazioni sulla configura zione e l'utilizzo delle credenziali.	22 febbraio 2021
the section called “Configurare un progetto GraalVM Native Image”	Nuovo argomento per la configurazione di un progetto GraalVM Native Image.	18 febbraio 2021
the section called “Sondaggio sugli stati delle risorse”	Waiters è stato rilasciato; è stato aggiunto un argomento per la nuova funzionalità.	30 settembre 2020
the section called “Pubblica le metriche SDK”	Metriche rilasciate; è stato aggiunto un argomento per la nuova funzionalità.	17 agosto 2020
the section called “Amazon SNS”	Sono stati aggiunti argomenti di esempio per Amazon SNS.	30 maggio 2020
the section called “Riduci i tempi di avvio dell'SDK per AWS Lambda”	È stato aggiunto un argomento sulle prestazioni delle AWS Lambda funzioni.	29 maggio 2020
the section called “Imposta il TTL JVM per le ricerche dei nomi DNS”	È stato aggiunto l'argomento di caching DNS JVM TTL.	27 aprile 2020
the section called “Configurare un progetto Apache Maven”, the section called “Configura un progetto Gradle”	Nuovi argomenti di configura zione di Maven e Gradle.	21 aprile 2020

Modifica	Descrizione	Data
the section called “Transport Layer Security (TLS)”	Aggiunto TLS 1.2 alla sezione sulla sicurezza.	19 marzo 2020
the section called “Iscrivit i a Amazon Kinesis Data Streams”	Aggiunti esempi Kinesis di stream.	2 agosto 2018
the section called “Lavora con risultati impaginati”	Aggiunto argomento paginazio ne automatica.	5 Aprile 2018
???	Sono stati aggiunti argomenti di esempio per IAM Amazon EC2, CloudWatch e DynamoDB.	29 dicembre 2017
the section called “Amazon S3”	È stato aggiunto l'esempio getobjects per Amazon S3.	7 agosto 2017
the section called “Usa la programmazione asincrona”	Aggiunto argomento asincrono .	4 agosto 2017
Versione GA della 2.x AWS SDK per Java	AWS SDK per 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.